Java Agents

June 30, 2025

JAVA AGENTS
Version: 2024.3

Table of Contents

Installation

This guide describes how to install PingAM Java Agent.

About Ping Identity Platform™ software

Ping 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:

- Java Agent installed on `http://agent.example.com:80`.
- PingAM installed on `http://am.example.com:8080/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 Java Agent in different containers.
- Install the container before you install the agent.
- Install only one Java Agent for each container.
- Install a supported version of the Java runtime environment, as described in Java requirements. Set the `JAVA_HOME` environment variable accordingly. The agent installer requires Java.

```
$ echo $JAVA_HOME
/path/to/java
```

- 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. For more information, refer to Configure load balancers and reverse proxies.

Download and unzip Java Agent

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:

Directory	Description
`bin`	The `agentadmin` installation and configuration program. Learn more from agentadmin command.
`config`	Configuration templates used by the `agentadmin` command during installation
`data`	Not used
`etc`	Configuration templates used during installation
`installer-logs`	Location of log files written during installation
`legal-notices`	Licensing information including third-party licenses
`lib`	Shared libraries used by the agent
`locale`	Property files used by the installation program
`README`	README file containing platform and install information for the agent

Preinstallation tasks

1. 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

```
C:> type > pwd.txt
password
CTRL+Z
```

In Windows Explorer, right-click the file, 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 using a password manager or by using the command line tool agentadmin --key.

2. (Optional) Create a signing key for pre-authentication cookies and POST data preservation cookies. The key must be at least 64 characters long, but preferably 80.

    1. Create the key with the agentadmin --key command:

Unix

```
$ agentadmin --key 80
ZRY...xXO
```

Windows

```
C:> agentadmin --key 80
ZRY...xXO
```

    2. Write the key to a file:

Unix

```
$ cat > /secure-directory/signing-key.txt
ZRY...xXO
CTRL+D

$ chmod 400 /secure-directory/signing-key.txt
```

Windows

```
C:> type > /secure-directory/signing-key.txt
ZRY...xXO
CTRL+Z
```

In Windows Explorer, right-click the file, select Read-Only, and then click OK.

3. In AM, add an agent profile, as described in Create agent profiles:

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

   - **Agent ID**: `java-agent`
   - **Agent URL**: `http://agent.example.com:80/app`
   - **Server URL**: `http://openam.example.com:8080/openam`
   - **Password**: `password`

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

   The examples in this guide use 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.

When you create your own policy set instead of using the default policy set, `iPlanetAMWebAgentService`, update the following properties in the agent profile:

- Policy Set Map
- Policy Evaluation Realm Map

5. When you exchange **signed** OpenID Connect JWTs between AM and the agent, set up a new key and secret as described in Configure Communication With AM Servers. Do not use the default `test` key pair in a real environment.

Configure communication with AM servers

AM communicates authentication and authorization information to Java Agent by using OpenID Connect (OIDC) JSON web tokens (JWT). To secure the JSON payload, AM and the agent support JWT signing with the RS256 algorithm. For more information, refer to RFC 7518.

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

By default, AM uses a demo key and an autogenerated secret for these purposes. For production environments, perform the steps in the following procedure 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 JWTs 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 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.

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

1. Create the following aliases in one of the secret stores configured in AM, for example, the default JCEKS keystore:
   - RSA key pair
   - HMAC secret
2. In the AM admin UI, select **Configure** > **Secret Stores** > Keystore Secret Store Name > **Mappings**, and configure the following secret IDs:
   - The new RSA key alias in the `am.global.services.oauth2.oidc.agent.idtoken.signing` secret ID.
   - The new HMAC secret in the `am.services.oauth2.jwt.authenticity.signing` secret ID.

   You might 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.

Create agent profiles

Java Agent requires a profile to connect to and communicate with AM, regardless of whether the agent is in remote configuration mode or local configuration mode.

This section describes how to create an agent profile and inherit properties from a group. Alternatively, create agent profiles by using the `/realm-config/agents/WebAgent/{id}` endpoint in the REST API.

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

Create an agent profile in the AM admin UI

1. In the AM admin UI, select **Realms** > realm name > **Applications** > **Agents** > **Java**, 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.

When the agent is in remote configuration mode, the Agent URL is used to populate the agent profile for services, such as notifications.

Server URL

The full URL to an authorization server, such as PingOne Advanced 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. When the agent is in remote configuration mode, the Server URL is used to populate the agent profile for login, logout, naming, and cross-domain SSO.

Password

The password the agent uses to authenticate to an authorization server, such as PingOne Advanced 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 using a password manager or by using the command line tool agentadmin --key.

2. (Optional - From AM 7.5) Use AM's secret service to manage the agent profile password. If AM finds a matching secret in a secret store, it uses that secret instead of the agent password configured in Step 1.

    1. In the agent profile page, set a label for the agent password in **Secret Label Identifier**.

        AM uses the identifier to generate a secret label for the agent.

        The secret label has the format `am.application.agents.identifier.secret`, where identifier is the **Secret Label Identifier**.

        The **Secret Label Identifier** can only contain characters `a-z`, `A-Z`, `0-9`, and periods (`.`). It can't start or end with a period.

    2. Select **Secret Stores** and configure a secret store.

    3. Map the label to the secret. Learn more from AM's mapping.

    Note the following points for using AM's secret service:

    - Set a **Secret Label Identifier** that clearly identifies the agent.
    - If you update or delete the **Secret Label Identifier**, AM updates or deletes the corresponding mapping for the previous identifier provided no other agent shares the mapping.
    - When you rotate a secret, update the corresponding mapping.

Create an agent profile with the `ssoadm` command line tool

For information about how to use `ssoadm` and properties with multiple aliases, see Property aliases.

For more information about `ssoadm`, refer to ssoadm in AM's *Reference*.

1. Set up `ssoadm`, as described in AM's Setting up administration tools in AM's *Installation*.
2. 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

```
C:> type > pwd.txt
password
CTRL+Z
```

In Windows Explorer, right-click the file, 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 using a password manager or by using the command line tool agentadmin --key.

3. Run the an `ssoadm` command similar to this to create the agent:

```
./ssoadm create-agent \
--agentname java-agent \
--agenttype J2EEAgent \
--password-file /secure-directory/pwd.txt \
--realm / \
--agenturl http://agent.example.com:80/app \
--serverurl http://am.example.com:8080/am \
--adminid uid=amadmin,ou=People,dc=am,dc=myorg,dc=org \
--attributevalues userpassword=password

Agent configuration was created.
```

4. (Optional) Configure additional properties for the agent, by adding them to the `--attributevalues` option.

   Add the following line to the above example to configure a value for Max Entries in Not-Enforced IP Cache:

```
--attributevalues com.sun.identity.agents.config.notenforced.ip.cache.size=2000
```

Create an agent profile group and inherit settings

Use agent profile groups to set up multiple agents that inherit settings from the group.

1. In the AM admin UI, select **Realms** > realm name > **Applications** > **Agents** > **Java**.
2. In the **Group** tab, add a group. Use the URL to the AM server in which to store the profile.
3. Edit the group configuration as necessary, and save the configuration.
4. Select **Realms** > realm name > **Applications** > **Agents** > **Java**, and select an agent you created previously.
5. In the **Global** tab, select **Group**, and add the agent to the group you created previously. The icon 🔒 appears next to some properties.

6. For each property where 🔒 appears, toggle the icon to set inheritance:
    - 🔓 Do not inherit the value from the group.
    - 🔒 Inherit the value from the group.

Authenticate agents to the identity provider

Authenticate agents to PingOne Advanced Identity Cloud

> **Important**
>
> Java Agent is automatically authenticated to PingOne Advanced Identity Cloud by a non-configurable authentication module. Authentication chains and modules are deprecated in PingOne Advanced Identity Cloud and replaced by journeys.
> You can now authenticate Java Agent to PingOne Advanced 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 PingOne Advanced Identity Cloud.
> For more information, refer to PingOne Advanced Identity Cloud's Journeys.

This section describes how to create a journey to authenticate Java Agent to PingOne Advanced 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 PingOne Advanced Identity Cloud, that same journey is used for all instances of PingGateway, Java Agent, and Web Agent. Consider this point if you change the journey configuration.

1. Log in to the Advanced 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 user or device to authenticate.
    - (Optional) **Description**: Authenticate an agent to PingOne Advanced Identity Cloud

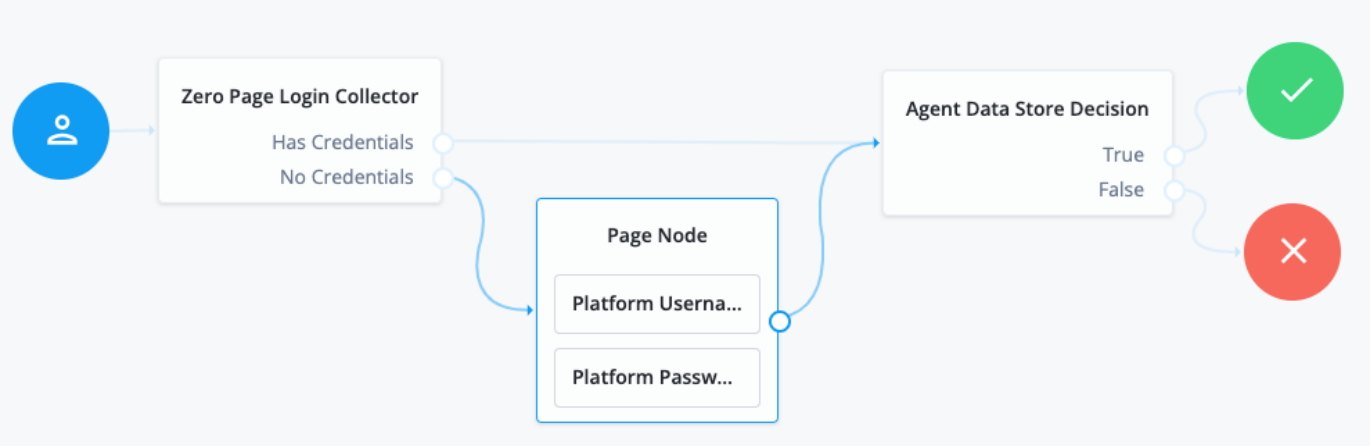
    The journey designer is displayed, with the `Start` entry point connected to the `Failure` exit point, and a `Success` node.
4. Using the **Filter nodes** bar, find and then drag the following nodes from the **Components** panel into the designer area:
    - Zero Page Login Collector node to check whether 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 Java Agent 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 to prompt the user to enter their username.
   - Platform Password node to prompt the user to enter their password.
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, Java Agent is automatically authenticated to AM by an authentication tree. Otherwise, Java Agent 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 Java Agent 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 PingGateway, 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:

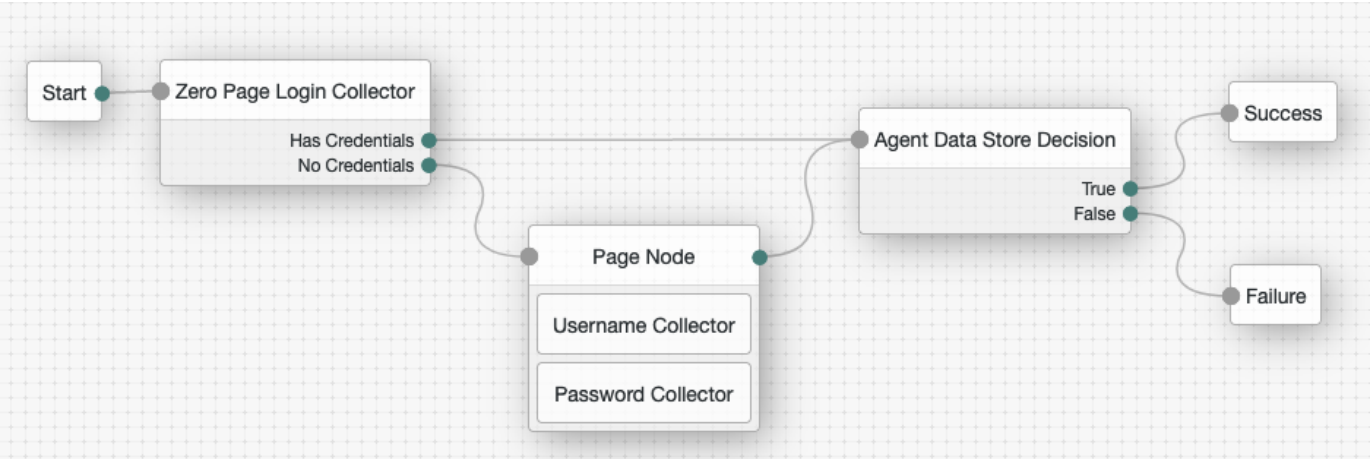
Button	Usage
	Lay out and align nodes according to the order they are connected.
	Toggle the designer window between normal and full-screen layout.
	Remove the selected node. Note that the `Start` entry point cannot be deleted.

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 Java 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:
   - 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:



Create agent administrators for a realm

To create agent profiles when installing Java Agent, you need the credentials of an AM user who can read and write agent profiles.

This section describes how to create an agent administrator for a specific realm. Use this procedure to reduce the scope given to users who create agent profiles.

1. In the AM admin UI, select **Realms** > realm name > **Identities**.
2. In the **Groups** tab, add a group for agent administrators.
3. In the **Privileges** tab, enable **Log Read** and **Log Write**.
4. Return to **Realms** > realm name > **Identities**, add agent administrator identities.
5. For each identity, select the **Groups** tab, add the user to agent profile administrator group.
6. Provide each system administrator who installs agents with their agent administrator credentials.

   When installing the agent with the `--custom-install` option, the system administrator can choose the option to create the profile during installation, and then provide the agent administrator username and the path to a read-only file containing the agent administrator password.

- For installation with PingOne Advanced Identity Cloud, install

Install Java Agent

Install Tomcat Java Agent

Before you install, make sure that all Tomcat scripts are present in the `$CATALINA_HOME/bin` directory. The Tomcat Windows executable installer does not include the scripts. If the scripts are not present in your installation, copy the contents of the `bin` directory from a `.zip` download of Tomcat of the same version as the one you installed.

Install Tomcat Java Agent interactively

1. Review the information in Before you install, and perform the steps in Preinstallation tasks.

2. Shut down the Tomcat server where you plan to install the agent.
3. Make sure AM is running.
4. Run `agentadmin --install` to install the agent:

```
$ /path/to/java_agents/tomcat_agent/bin/agentadmin --install
```

5. 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 Tomcat configuration folder:

```
...
[ ? : Help, ! : Exit ]
Enter the Tomcat Server Config Directory Path
[/opt/apache-tomcat/conf]: /path/to/apache-tomcat/conf
```

1. Enter the AM URL:

```
...
[ ? : Help, < : Back, ! : Exit ]
AM server URL: https://openam.example.com:8443/openam
```

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

If a reverse proxy is configured between AM and the agent, enter the proxy URL. For more information, refer to Configure an Apache HTTP Server as a reverse proxy.

2. Enter the `$CATALINA_HOME` environment variable, specifying the path to the root of the Tomcat server:

```
...
[ ? : Help, < : Back, ! : Exit ]
Enter the $CATALINA_HOME environment variable: /path/to/apache-tomcat
```

3. Enter the agent URL:

```
...
[ ? : Help, < : Back, ! : Exit ]
Agent URL: \http://agent.example.com:80
```

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

```
...
[ ? : Help, < : Back, ! : Exit ]
Enter the Agent Profile name: java-agent
```

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

```
...
[ ? : Help, < : Back, ! : Exit, ^ : Accept Empty value ]
Enter the Agent Profile realm [/]:
```

6. Enter the path to the password file you created during pre-installation:

```
...
[ ? : Help, < : Back, ! : Exit ]
Enter the path to the password file: /secure-directory/pwd.txt
```

7. Enter the path to a file containing the agent pre-authentication cookie signing value:

```
...
[ ? : Help, < : Back, ! : Exit ]
Enter the path to the signing file:
```

Provide a path to a file containing a randomly generated key that is at least 64 characters long but preferably about 80 characters. For help to create signing a key, refer to Create a cookie signing key.

For information about how the agent uses pre-authentication cookies, refer to the *Authentication* section of Request flow.

To disable cookie signing, press return without providing a value.

> **Tip**
>
> Cookie signing is a CPU-intensive process that renders cookies more tamper-proof. Weigh the potential increase in security against the potential loss in performance.

6. Review a summary of your responses and select how to continue:

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

After successful installation, the installer adds the agent configuration to the Tomcat configuration, and sets up configuration and log directories for the agent.

7. Test the installation by browsing to a resource that the agent protects. AM redirects you to authenticate. After authentication, AM redirects you back to the requested resource.

Install Tomcat Java Agent silently

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

The following example uses a response file containing the same configuration as in Install Tomcat Java Agent interactively.

1. Review the information in Before you install, and perform the steps in Preinstallation tasks.
2. Shut down the Tomcat server where you plan to install the agent.
3. Make sure AM is running.
4. Create a response file with the following content, at `/path/to/response-file`:

```
# Response File
CONFIG_DIR= /path/to/apache-tomcat/conf
AM_SERVER_URL= https://am.example.com:8443/am
CATALINA_HOME= /path/to/apache-tomcat
AGENT_URL= \http://agent.example.com:80
AGENT_PROFILE_NAME= java-agent
AGENT_PROFILE_REALM= /
AGENT_PASSWORD_FILE= /secure-directory/pwd.txt
AGENT_SIGNING_FILE= /secure-directory/signing-key.txt
```

5. Run the `agentadmin` command with the `--useResponse` option:

```
$ agentadmin --install --useResponse /path/to/response-file
```

Install in a subrealm

Other installation examples 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_PROFILE_REALM = /
```

specify:

```
AGENT_PROFILE_REALM = /myrealm
```

Even though the agent is installed in a subrealm, the default login redirect requires users to log into the top-level realm. For information about how to change the login, refer to Use the request domain to redirect login to a different realm.

Install JBoss Java Agent

The examples in this section assume that you are using JBoss, but the procedures are the same for WildFly. Agent binaries for JBoss and WildFly are the same.

Install JBoss Java Agent interactively

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

   ```
   $ /path/to/java_agents/jboss_agent/bin/agentadmin --install
   ```

5. Enter the absolute path to the JBoss installation directory:

   ```
   ...
   [ ? : Help, ! : Exit ]
   Enter the path to the JBoss installation: /path/to/jboss
   ```

6. Enter the name of the deployment mode for the JBoss installation:
   - `standalone`: Manage a single JBoss instance

     In standalone mode, the agent installer uses an auto-deployment feature provided by the JBoss deployment scanner so that you do not have to deploy the `agentapp.war` manually.
   - `domain`: Manage multiple server instances from a single control point.

     In this mode, at the end of the procedure, you must manually deploy the `java_agents/jboss_agent/etc/agentapp.war` file to JBoss.
7. Enter the name of the profile to use in `standalone` or `domain` mode:
   - `standalone`: Default.
   - `full`: Supports Java EE 6 Full Profile, and subsystems that are not required for high-availability.
   - `ha`: Enables all default subsystems, and adds the clustering capabilities.
   - `full-ha`: Enables all default subsystems, including those required for high-availability, and adds clustering capabilities.
8. Choose whether to deploy the agent as a global JBoss module.

```
...
[ ? : Help, < : Back, ! : Exit ]
Install agent as global module? [true]: true
```

To include specific modules for a web application, enter `false`, and complete the additional steps at the end of this procedure.

9. Enter the AM URL, including the deployment URI:

```
...
[ ? : Help, < : Back, ! : Exit ]
AM server URL: https://am.example.com:8443/am
```

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

If a reverse proxy is configured between AM and the agent, enter the proxy URL. For more information, refer to Configure an Apache HTTP Server as a reverse proxy.

10. Enter the agent URL:

```
...
[ ? : Help, < : Back, ! : Exit ]
Agent URL: \http://agent.example.com:80
```

11. Enter the agent profile name created in AM as part of the pre-installation procedure:

```
...
[ ? : Help, < : Back, ! : Exit ]
Enter the Agent Profile name: JBossAgent
```

12. Enter the realm in which the specified agent profile exists.

Press ENTER to accept the default value of `/` for the top-level realm. If you specify the `(^) : Accept Empty value` option, the top-level realm is used.

```
...
[ ? : Help, < : Back, ! : Exit, ^ : Accept Empty value ]
Enter the Agent Profile realm [/]:
```

13. Enter the path to the password file you created as part of the pre-installation procedure:

```
...
[ ? : Help, < : Back, ! : Exit ]
Enter the path to the password file: /secure-directory/pwd.txt
```

1. Enter the path to a file containing the agent pre-authentication cookie signing value:

   ```
   ...
   [ ? : Help, < : Back, ! : Exit ]
   Enter the path to the signing file:
   ```

   Provide a path to a file containing a randomly generated key that is at least 64 characters long but preferably about 80 characters. For help to create signing a key, refer to Create a cookie signing key.

   For information about how the agent uses pre-authentication cookies, refer to the *Authentication* section of Request flow.

   To disable cookie signing, press return without providing a value.

   > **Tip**
   >
   > Cookie signing is a CPU-intensive process that renders cookies more tamper-proof. Weigh the potential increase in security against the potential loss in performance.

14. Review a summary of your responses and select how to continue:

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

    After successful completion, the installer updates the JBoss configuration, adds the agent web application under `JBOSS_HOME/server/standalone/deployments`, and sets up configuration and log directories for the agent.

15. Follow these steps if you responded `false` to the question `Deploy the policy agent as a global JBoss module` during the installation:

    1. Add the following line to the web application file `/path/to/protected/app/META-INF/MANIFEST.MF`:

       ```
       Dependencies: org.forgerock.openam.agent
       ```

    2. Create a file at `/path/to/protected/app/WEB-INF/jboss-deployment-structure.xml` with the following content:

```
<?xml version="1.0"?>
 <jboss-deployment-structure xmlns="urn:jboss:deployment-structure:1.2" xmlns:xsi="http://www.w3.org/
2001/XMLSchema-instance">
  <deployment>
   <dependencies>
    <module name="org.forgerock.openam.agent" >
     <imports>
      <include path="META-INF"/>
      <include path="org"/>
     </imports>
    </module>
   </dependencies>
  </deployment>
</jboss-deployment-structure>
```

16. If you chose `domain` as the deployment mode, manually deploy the `java_agents/jboss_agent/etc/agentapp.war` file to JBoss.
17. Test the installation by browsing to a resource that the agent protects. AM redirects you to authenticate. After authentication, AM redirects you back to the requested resource.

Install JBoss Java Agent Silently

To install the Java Agent silently, create a response file containing the installation parameters, and then provide it to the `agentadmin` command.

The following is an example response file to install the agent when JBoss is configured in `standalone` mode:

```
# Agent User Response File
HOME_DIR= /path/to/jboss
INSTANCE_NAME= standalone
GLOBAL_MODULE= true
INSTALL_PROFILE_NAME=
AM_SERVER_URL= https://am.example.com:8443/am
AGENT_URL= http://www.example.com:8080/agentapp
AGENT_PROFILE_NAME= JBossAgent
AGENT_PROFILE_REALM= /
AGENT_PASSWORD_FILE= /secure-directory/pwd.txt
AGENT_SIGNING_FILE= /secure-directory/signing-key.txt
```

The `INSTALL_PROFILE_NAME` variable is used only when the `INSTANCE_NAME` is set to `domain`. It specifies the name of the JBoss domain profile.

To load balance connections between the agent and an AM site, set `AM_SERVER_URL` to the URL of the load balancer in front of the AM site.

If a reverse proxy is configured between AM and the agent, set `AM_SERVER_URL` to the proxy URL. For more information, refer to Configure an Apache HTTP Server as a reverse proxy.

1. Review the information in Before you install, and perform the steps in Preinstallation tasks.
2. Make sure that the response file for the installation is ready, or create a response file, for example:

```
$ agentadmin --install --saveResponse response-file
```

3. Shut down the JBoss server where you plan to install the agent.
4. Make sure AM is running.
5. Run the `agentadmin` command with the `--useResponse` option:

```
$ agentadmin --install --useResponse /path/to/response-file
```

6. If you configured the `GLOBAL_MODULE` variable as `false` in the response file, add the following line to the `META-INF/MANIFEST.MF` file of the web application:

```
Dependencies: org.forgerock.openam.agent
```

7. If you configured the `INSTANCE_NAME` variable as `domain` in the response file, manually deploy the `java_agents/jboss_agent/etc/agentapp.war` file to JBoss.

Install in a subrealm

Other installation examples 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_PROFILE_REALM = /
```

specify:

```
AGENT_PROFILE_REALM = /myrealm
```

Even though the agent is installed in a subrealm, the default login redirect requires users to log into the top-level realm. For information about how to change the login, refer to Use the request domain to redirect login to a different realm.

Install Jetty Java Agent

Consider the following points before you install:

- For installation on Jetty 12, you can use Javax EE8, Jakarta EE9, or Jakarta EE10. Make sure you use the correct agent (Javax or Jakarta) for your environment.
- Command-line examples in this chapter show Jetty accessed remotely. If you have issues accessing Jetty remotely, consider changing the filter settings in the deployment descriptor file, `/path/to/jetty/webapps/test/WEB-INF/web.xml`, as shown in the following example:

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

Jetty configuration

The Jetty compliance mode is configured by default to prevent path traversal vulnerabilities. Consider the impact on security before you change org.eclipse.jetty.http.UriCompliance to a less safe value such as `UNSAFE` or `RFC3986`.

Install Jetty Java Agent interactively

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

   ```
   $ /path/to/java_agents/jetty_agent/bin/agentadmin --install
   ```

5. Enter the absolute path to the root of the Jetty installation:

   ```
   ...
   [ ? : Help, ! : Exit ]
   Enter the Jetty home directory [/opt/jetty]: /path/to/jetty/home
   ```

   This is the same as the `JETTY_HOME` environment variable for Jetty.

6. Enter the absolute path to the Jetty configuration directory:

   ```
   ...
   [ ? : Help, &lt; : Back, ! : Exit ]
   Enter the absolute path of the Jetty etc directory: /path/to/jetty/etc
   ```

7. Enter the absolute path to the Jetty base directory:

   ```
   ...
   [ ? : Help, < : Back, ! : Exit ]
   Enter the Jetty base directory [/usr/local/jetty]: /path/to/jetty/base
   ```

   This is the same as the `JETTY_BASE` environment variable for Jetty.

This path may be the same as the one specified as the root of the Jetty installation.

8. Enter the AM URL, including the deployment URI:

```
...
[ ? : Help, < : Back, ! : Exit ]
AM server URL: https://am.example.com:8443/am
```

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

If a reverse proxy is configured between AM and the agent, enter the proxy URL. For more information, refer to Configure an Apache HTTP Server as a reverse proxy.

9. Enter the agent URL:

```
...
[ ? : Help, < : Back, ! : Exit ]
Agent URL: \http://agent.example.com:80
```

10. Enter the agent profile name created in AM as part of the pre-installation procedure:

```
...
[ ? : Help, &lt; : Back, ! : Exit ]
Enter the Agent Profile name: JettyAgent
```

11. Enter the realm in which the specified agent profile exists.

Press ENTER to accept the default value of `/` for the top-level realm. If you specify the `(^) : Accept Empty value` option, the top-level realm is used.

```
...
[ ? : Help, < : Back, ! : Exit, ^ : Accept Empty value ]
Enter the Agent Profile realm [/]:
```

12. Enter the path to the password file you created as part of the pre-installation procedure:

```
...
[ ? : Help, < : Back, ! : Exit ]
Enter the path to the password file: /secure-directory/pwd.txt
```

    1. Enter the path to a file containing the agent pre-authentication cookie signing value:

```
...
[ ? : Help, < : Back, ! : Exit ]
Enter the path to the signing file:
```

Provide a path to a file containing a randomly generated key that is at least 64 characters long but preferably about 80 characters. For help to create signing a key, refer to Create a cookie signing key.

For information about how the agent uses pre-authentication cookies, refer to the *Authentication* section of Request flow.

To disable cookie signing, press return without providing a value.

> **Tip**
>
> Cookie signing is a CPU-intensive process that renders cookies more tamper-proof. Weigh the potential increase in security against the potential loss in performance.

13. Review a summary of your responses and select how to continue:

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

After successful completion, the installer updates Jetty's `start.jar` to refer to the agent, sets up the agent web application, and sets up configuration and log directories for the agent.

14. Test the installation by browsing to a resource that the agent protects. AM redirects you to authenticate. After authentication, AM redirects you back to the requested resource.

Install Jetty Java Agent silently

To install the Java Agent silently, create a response file containing the installation parameters, and then provide it to the `agentadmin` command. The following is an example response file:

```
# Agent User Response File
CONFIG_DIR= /path/to/jetty/etc
JETTY_HOME= /path/to/jetty/home
JETTY_BASE= /path/to/jetty/base
AM_SERVER_URL= https://am.example.com:8443/am
AGENT_URL= http://www.example.com:8080/agentapp
AGENT_PROFILE_NAME= JettyAgent
AGENT_PROFILE_REALM= /
AGENT_PASSWORD_FILE= /secure-directory/pwd.txt
AGENT_SIGNING_FILE= /secure-directory/signing-key.txt
```

To load balance connections between the agent and an AM site, set `AM_SERVER_URL` to the URL of the load balancer in front of the AM site.

If a reverse proxy is configured between AM and the agent, set `AM_SERVER_URL` to the proxy URL. For more information, refer to Configure an Apache HTTP Server as a reverse proxy.

1. Review the information in Before you install, and perform the steps in Preinstallation tasks.
2. Shut down the Jetty server where you plan to install the agent.
3. Make sure that AM is running.
4. Run the `agentadmin` command with the `--useResponse` option:

```
$ agentadmin --install --useResponse /path/to/response-file
```

Install in a subrealm

Other installation examples 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_PROFILE_REALM = /
```

specify:

```
AGENT_PROFILE_REALM = /myrealm
```

Even though the agent is installed in a subrealm, the default login redirect requires users to log into the top-level realm. For information about how to change the login, refer to Use the request domain to redirect login to a different realm.

Install WebLogic Java Agent

Install WebLogic Java Agent interactively

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

```
$ /path/to/java_agents/weblogic_agent/bin/agentadmin --install
```

5. Enter the path to the `startWebLogic.sh` file of the WebLogic domain where you want to install the agent:

```
...
[ ? : Help, ! : Exit ]
Enter the Startup script location
[/usr/local/bea/user_projects/domains/base_domain/startWebLogic.sh]:
/path/to/Oracle_Home/user_projects/domains/base_domain/startWebLogic.sh
```

6. Enter the path to the WebLogic installation directory:

```
...
[ ? : Help, < : Back, ! : Exit ]
Enter the WebLogic home directory [/usr/local/bea/wlserver_10.0]:
/path/to/weblogic
```

7. Enter the AM URL, including the deployment URI:

```
...
[ ? : Help, < : Back, ! : Exit ]
AM server URL: https://am.example.com:8443/am
```

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

   If a reverse proxy is configured between AM and the agent, enter the proxy URL. For more information, refer to Configure an Apache HTTP Server as a reverse proxy.

8. Enter the agent URL:

```
...
[ ? : Help, < : Back, ! : Exit ]
Agent URL: \http://agent.example.com:80
```

9. Enter the agent profile name created in AM as part of the pre-installation procedure:

```
…
[ ? : Help, < : Back, ! : Exit ]
Enter the Agent Profile name: WebLogicAgent
```

10. Enter the realm in which the specified agent profile exists.

    Press ENTER to accept the default value of `/` for the top-level realm. If you specify the `(^) : Accept Empty value` option, the top-level realm is used.

```
...
[ ? : Help, < : Back, ! : Exit, ^ : Accept Empty value ]
Enter the Agent Profile realm [/]:
```

11. Enter the path to the password file you created as part of the pre-installation procedure:

```
...
[ ? : Help, < : Back, ! : Exit ]
Enter the path to the password file: /secure-directory/pwd.txt
```

    1. Enter the path to a file containing the agent pre-authentication cookie signing value:

```
...
[ ? : Help, < : Back, ! : Exit ]
Enter the path to the signing file:
```

    Provide a path to a file containing a randomly generated key that is at least 64 characters long but preferably about 80 characters. For help to create signing a key, refer to Create a cookie signing key.

    For information about how the agent uses pre-authentication cookies, refer to the *Authentication* section of Request flow.

    To disable cookie signing, press return without providing a value.

> **Tip**
>
> Cookie signing is a CPU-intensive process that renders cookies more tamper-proof. Weigh the potential increase in security against the potential loss in performance.

12. Review a summary of your responses and select how to continue:

```
$ /path/to/java_agents/weblogic_agent/bin/agentadmin --install

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

13. Source the agent in one of the following ways:

    - Manually source the file containing the agent environment settings for WebLogic before starting the container.

```
$ . /path/to/setAgentEnv_AdminServer.sh
```

    - Add the `setAgentEnv_AdminServer.sh` line to the shown location [path] in the `startWebLogic.sh` script. Note that the file can be overwritten:

```
$ cat /path/to/startWebLogic.sh
...
# Any changes to this script may be lost when adding extensions to this
# configuration.
DOMAIN_HOME="/opt/Oracle/Middleware/user_projects/domains/base_domain"
. /path/to/setAgentEnv_AdminServer.sh
${DOMAIN_HOME}/bin/startWebLogic.sh $*
```

If the sourcing is not set properly, the following message appears:

```
<Error> <HTTP> <cent.example.com>
<AdminServer> <[STANDBY] ExecuteThread: '5' for queue: weblogic.kernel.
Default (self-tuning)'> <<WLS Kernel>>
<BEA-101165> <Could not load user defined filter in web.xml:
ServletContext@1761850405[app:agentapp module:agentapp.war path:null
spec-version:null] com.sun.identity.agents.filter.AmAgentFilter.
java.lang.ClassNotFoundException:
com.sun.identity.agents.filter.AmAgentFilter
```

14. Start the WebLogic server.
15. Deploy the `/path/to/java_agents/weblogic_agent/etc/agentapp.war` agent web application in WebLogic.
16. Test the installation by browsing to a resource that the agent protects. AM redirects you to authenticate. After authentication, AM redirects you back to the requested resource.

Install WebLogic Java Agent silently

To install the Java Agent silently, create a response file containing the installation parameters, and then provide it to the `agentadmin` command. The following is an example response file:

```
# Agent User Response File
STARTUP_SCRIPT= /path/to/Oracle_Home/user_projects/domains/base_domain/startWebLogic.sh
SERVER_NAME= AdminServer
WEBLOGIC_HOME_DIR= /path/to/weblogic
AM_SERVER_URL= https://am.example.com:8443/am
AGENT_URL= http://www.example.com:8080/agentapp
AGENT_PROFILE_NAME= WebLogicAgent
AGENT_PROFILE_REALM= /
AGENT_PASSWORD_FILE= /secure-directory/pwd.txt
AGENT_SIGNING_FILE= /secure-directory/signing-key.txt
```

To load balance connections between the agent and an AM site, set `AM_SERVER_URL` to the URL of the load balancer in front of the AM site.

If a reverse proxy is configured between AM and the agent, set `AM_SERVER_URL` to the proxy URL. For more information, refer to Configure an Apache HTTP Server as a reverse proxy.

1. Review the information in Before you install, and perform the steps in Preinstallation tasks.
2. Make sure that the response file for the installation is ready, or create a response file, for example:

```
$ agentadmin --install --saveResponse response-file
```

3. Shut down the WebLogic server where you plan to install the agent.
4. Make sure AM is running.
5. Run the `agentadmin` command with the `--useResponse` option:

```
$ agentadmin --install --useResponse /path/to/response-file
```

6. Source the agent in one of the following ways:
   - Manually source the file containing the agent environment settings for WebLogic before starting the container.

     ```
     $ . /path/to/setAgentEnv_AdminServer.sh
     ```

   - Add the `setAgentEnv_AdminServer.sh` line to the shown location [path] in the `startWebLogic.sh` script. Note that the file can be overwritten:

     ```
     $ cat /path/to/startWebLogic.sh
     ...
     # Any changes to this script may be lost when adding extensions to this
     # configuration.
     DOMAIN_HOME="/opt/Oracle/Middleware/user_projects/domains/base_domain"
     . /path/to/setAgentEnv_AdminServer.sh
     ${DOMAIN_HOME}/bin/startWebLogic.sh $*
     ```

     If the sourcing is not set properly, the following message appears:

     ```
     <Error> <HTTP> <cent.example.com>
     <AdminServer> <[STANDBY] ExecuteThread: '5' for queue: weblogic.kernel.
     Default (self-tuning)'> <<WLS Kernel>>
     <BEA-101165> <Could not load user defined filter in web.xml:
     ServletContext@1761850405[app:agentapp module:agentapp.war path:null
     spec-version:null] com.sun.identity.agents.filter.AmAgentFilter.
     java.lang.ClassNotFoundException:
     com.sun.identity.agents.filter.AmAgentFilter
     ```

7. Start the WebLogic Server.
8. Deploy the `/path/to/java_agents/weblogic_agent/etc/agentapp.war` agent web application in WebLogic.

Install WebLogic Java Agent in multi-server domains

In many WebLogic domains, the administration server provides a central point for controlling and managing the configuration of the managed servers that host protected web applications.

If WebLogic-managed servers run on different hosts, you must create separate agent profiles and perform separate installations for each so that AM can send notifications to the appropriate addresses.

Install WebLogic Java Agent on administration and managed servers

1. If servers are on different hosts, create agent profiles for each server where you plan to install the agent. For more information, refer to Installing the WebLogic Java Agent.
2. Prepare your protected web applications by adding the agent filter configuration as described in Configure the agent filter for a web application.
3. Use the `agentadmin` command to install the agent either interactively, or silently on each server in the domain:
    - For interactive installation, follow the instructions in To install the WebLogic Java Agent.
    - For silent installation, follow the instructions in Installing the WebLogic Java Agent silently.
4. On each managed server in the domain, update the classpath to include agent .jar files.

    In WebLogic Node Manager console, navigate to Environment > Servers > server > **Server Start** > **Class Path**, and then edit the classpath as in the following example, but all on a single line:

    ```
    /path/to/java_agents/weblogic_agent/lib/agent.jar:
    /path/to/java_agents/weblogic_agent/lib/openssoclientsdk.jar:
     ...
    /path/to/java_agents/weblogic_agent/locale:
    /path/to/java_agents/weblogic_agent/Agent_001/config:
    $CLASSPATH
    ```

    Replace the paths in the example with the actual paths for your domain.
5. Restart the managed servers.

Post-installation tasks

Review directories for configuration, logs, and POST data.

Each agent instance has a numbered directory, starting with `Agent_001` for the first instance. The following directories are created under `/path/to/java_agents/agent_type/Agent_n`:

- `config`: Learn more from Agent configuration.
- `logs`: During agent startup, the location of the logs is based on the container which is being used. For example, bootstrap logs for Tomcat agents are written to `catalina.out`. The following log directories are created:
    - `logs/audit/`: Operational audit log directory, used only if remote logging to AM is disabled.
    - `logs/debug/`: The directory where the agent writes debug log files after startup.
- `pdp`: The directory to store POST data. The directory is created on installation, but used only when Enable POST Data Preservation and POST Data Preservation in Files or Cache are `true`.

Configure the agent filter for a web application

After installation, configure an *agent filter* to intercept inbound client requests and give them access to resources. The agent filter class is `com.sun.identity.agents.filter.AmAgentFilter`. The agent filter gives access based on the value of Agent Filter Mode Map.

Configure the agent filter in the web application's `web.xml` file. For information about configuration options, refer to the documentation for your web application. For example, refer to Oracle's Developing Web Applications for WebLogic Server.

Configure the agent filter first, before configuring other filters in `web.xml`. If several web applications run in the same container, configure an agent filter for each web application.

The following example protects every resource in the web application where it is configured:

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

The following example protects an application that processes requests asynchronously:

```
<filter>
  <filter-name>Agent</filter-name>
  <display-name>AM Agent</display-name>
  <description>AM Agent Filter</description>
  <filter-class>com.sun.identity.agents.filter.AmAgentFilter</filter-class>
  <async-supported>true</async-supported>
</filter>
```

Configure the agent filter mode

By default, the agent filter uses the filter mode `URL_POLICY`. After installation, you can change the filter mode with the property Agent Filter Mode Map, or in the AM admin UI:

1. In the AM admin UI, go to **Realms** > realm name > **Application** > **Agents** > **Java**, and select your Java Agent.
2. On the **Global** tab, select **Agent Filter Mode Map**, and set the filter mode as follows:
    - To use `URL_POLICY` for all web applications in the web container, do not change the setting; this is the default.
    - To use `SSO_ONLY` for the `BankApp` web application, set these values:

3. (Optional) In **Agent Filter Mode**, override the global mode for a specific context path:
    - **Key**: `BankApp`.
    - **Value**: Enter the mode name, for example `URL_POLICY`.
4. Click **Add**, and save your changes.

Configure strategy for AM downtime

By default, if AM becomes unavailable at runtime, for example, due to network errors, the agent responds as follows:

- Matches incoming requests against not-enforced rules
- Resolves unmatched requests against the policy cache
- Returns HTTP 503 for requests that don't match cached results

The cache expires after the time defined by Policy Cache TTL.

To change the agent response to AM downtime, configure Strategy when AM unavailable.

Secure connections

Secure communication between the agent and AM

After installation, consider securing communication between the agent and AM.

1. Configure AM to send cookies only when the communication channel is secure:
    1. In the AM admin UI, select **Realms** > realm name > **Application** > **Agents** > **Java** > agent name > **SSO**.
    2. Enable Transmit Cookies Securely.
2. Import a CA certificate in the JDK truststore, usually at `$JAVA_HOME/jre/lib/security/cacerts`. The certificate should be the one configured for HTTPS connections in the AM container, or signed with the same CA root certificate. For example:

    ```
    $ keytool \
    -import \
    -trustcacerts \
    -alias agentcert \
    -file /path/to/cacert.pem \
    -keystore $JAVA_HOME/jre/lib/security/cacerts
    ```

    Make sure that all containers where AM is installed trust the certificate stored in the JDK truststore, and that the JDK trusts the certificates stored on the containers where AM is installed.
3. Add the following properties to the `AgentBootstrap.properties` file:
    - `javax.net.ssl.trustStore`, to specify the full path to the JDK truststore.
    - `javax.net.ssl.trustStorePassword`, to specify the password of the truststore.

For example:

```
javax.net.ssl.trustStore=/Library/Java/JavaVirtualMachines/jdk1.8.0_101.jdk/Contents/Home/jre/lib/
security/cacerts
javax.net.ssl.trustStorePassword=changeit
```

For backward-compatibility, you can also provide the truststore and the password to the agent by specifying them as Java properties in the container's start-up sequence. For example, add them to Tomcat's `$CATALINA_OPS` variable instead of specifying them in the `AgentBootstrap.properties` file:

```
$ export CATALINA_OPTS="$CATALINA_OPTS \
-Djavax.net.ssl.trustStore=$JAVA_HOME/jre/lib/security/cacerts \
-Djavax.net.ssl.trustStorePassword=changeit"
```

4. Restart the agent.

Integrate with Bouncy Castle FIPS provider

This section gives an example of how to use the Bouncy Castle FIPS provider. For more information, refer to JAVA FIPS RESOURCES. The example uses Tomcat Java Agent but you can adapt it for other agent types.

Perform this procedure before installing the agent and starting the container.

1. Download the latest version of Bouncy Castle FIPS library from JAVA FIPS RESOURCES. This example uses `bc-fips-1.0.2.3.jar`.
2. Copy the .jar file to the agent library:
    1. Using the `.amAgentLocator` file, find the directory in which the agent is installed. In this example, the agent is installed in `/path/to/java_agents/tomcat_agent`:

Unix

```
$ cd /path/to/tomcat
$ cat .amAgentLocator; echo

/path/to/java_agents/tomcat_agent
```

Windows

```
C:\opt\container> type .amAgentLocator

C:/path/to/java_agents/tomcat_agent
```

2. Copy `bc-fips-1.0.2.3.jar` to the `lib` subdirectory:

Unix

```
$ cd /path/to/java_agents/tomcat_agent/lib
$ cp /tmp/bc-fips-1.0.2.3.jar /path/to/java_agents/tomcat_agent/lib
```

Windows

```
C:\opt\container> cd C:\path\to\java_agents\tomcat_agent\lib
C:\path\to\java_agents\tomcat_agent\lib> copy C:\temp{fips-jar} .
```

3. Set up the security providers to use Bouncy Castle:

    1. Locate the `java.security` file for your Java instance. For example, in Java 17 and Ubuntu the file is `/etc/java-17-openjdk/security/java.security`.

    2. Edit the file to place the `org.bouncycastle.jcajce.provider.BouncyCastleFipsProvider` line at the top of the list:

```
security.provider.1=org.bouncycastle.jcajce.provider.BouncyCastleFipsProvider
security.provider.2=SUN
security.provider.3=SunRsaSign
security.provider.4=SunEC
security.provider.5=SunJSSE
security.provider.6=SunJCE
security.provider.7=SunJGSS
security.provider.8=SunSASL
security.provider.9=XMLDSig
security.provider.10=SunPCSC
security.provider.11=JdkLDAP
security.provider.12=JdkSASL
security.provider.13=SunPKCS11
```

4. In the agent configuration, set `org.bouncycastle.fips.approved_only` to `true` so that only algorithms approved by FIPS can be used:

    1. Locate the `agentadmin` file:

    Unix

    ```
    $ cd /path/to/java_agents/tomcat_agent/bin
    ```

    Windows

    ```
    C:\> cd C:\path\to\java_agents\tomcat_agent\bin
    ```

    2. Change the following line:

    ```
    AGENT_OPTS="$AGENT_OPTS -Dagent.config.dir=$AGENT_HOME"
    ```

    to this line:

    ```
    AGENT_OPTS="$AGENT_OPTS -Dagent.config.dir=$AGENT_HOME -Dorg.bouncycastle.fips.approved_only=true"
    ```

5. Configure the Tomcat container to use the BouncyCastle provider. There are many ways to configure the container; this example uses a `setenv.sh` file:

    1. Locate or create a `setenv.sh` file for your Tomcat container. When Tomcat installed in `/path/to/tomcat/`, the file can be `/path/to/tomcat/bin/setenv.sh`.

    2. Add the following line for the `bc-fips-1.0.2.3.jar` classpath:

    ```
    CLASSPATH=/path/to/bc-fips-1.0.2.3.jar
    ```

    3. Add the following line to run the FIPS module in approved mode:

    ```
    JAVA_OPTS="$JAVA_OPTS -Dorg.bouncycastle.fips.approved_only=true"
    ```

    4. (Optional) Add the following property to the JAVA_OPTS to enable logs:

```
-Djava.security.debug=jca
```

6. Install the agent and start the container, as described in Install Tomcat Java Agent.

Remove Java Agent

Remove Tomcat Java Agent

1. Shut down the server where the agent is installed.
2. Run the `agentadmin` command with the `--listAgents` option list installed agent instances:

```
$ agentadmin --listAgents
The following agents are configured on this Application Server.
...
The following are the details for agent Agent_001 :-
...
```

3. Note the configuration information of the agent instance you want to remove.
4. Run the `agentadmin` command with the `--uninstall` option.

```
$ agentadmin --uninstall
```

5. Enter the path of the Tomcat installation directory:

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

6. Review a summary of your responses and select how to continue:

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

Remove JBoss Java Agent

1. Shut down the server where the agent is installed.
2. Run the `agentadmin` command with the `--listAgents` option list installed agent instances:

```
$ agentadmin --listAgents
The following agents are configured on this Application Server.
...
The following are the details for agent Agent_001 :-
...
```

3. Note the configuration information of the agent instance you want to remove.
4. Run the `agentadmin` command with the `--uninstall` option.

```
$ agentadmin --uninstall
```

5. Enter the path to the JBoss installation directory:

```
Enter the complete path to the home directory of the JBoss instance.
[ ? : Help, ! : Exit ]
Enter the path to the JBoss installation: /path/to/jboss
```

6. Enter `domain` or `standalone`, for the deployment mode of the JBoss installation to uninstall:

```
Enter the name of the deployment mode of the JBoss installation that you wish
to use with this agent. Supported values are: domain, standalone.
[ ? : Help, < : Back, ! : Exit ]
Enter the deployment mode of JBoss [standalone]: standalone
```

7. Review a summary of your responses and select how to continue:

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

Remove Jetty Java Agent

1. Shut down the server where the agent is installed.
2. Run the `agentadmin` command with the `--listAgents` option list installed agent instances:

```
$ agentadmin --listAgents
The following agents are configured on this Application Server.
...
The following are the details for agent Agent_001 :-
...
```

3. Note the configuration information of the agent instance you want to remove.
4. Run the `agentadmin` command with the `--uninstall` option.

```
$ agentadmin --uninstall
```

5. Enter the path of the Jetty configuration directory:

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

6. Review a summary of your responses and select how to continue:

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

Remove WebLogic Java Agent

1. Shut down the server where the agent is installed.
2. Run the `agentadmin` command with the `--listAgents` option list installed agent instances:

```
$ agentadmin --listAgents
The following agents are configured on this Application Server.
...
The following are the details for agent Agent_001 :-
...
```

3. Note the configuration information of the agent instance you want to remove.
4. Run the `agentadmin` command with the `--uninstall` option.

```
$ agentadmin --uninstall
```

5. Enter the path to the `startWebLogic.sh` file of the WebLogic domain where you want to install the agent:

```
Enter the path to the location of the script used to start the WebLogic domain.
Please ensure that the agent is first installed on the admin server instance
before installing on any managed server instance.
[ ? : Help, ! : Exit ]
Enter the Startup script location
[/usr/local/bea/user_projects/domains/base_domain/startWebLogic.sh]: /Oracle_Home/user_projects/domains/
base_domain/startWebLogic.sh
```

6. Enter the name of the WebLogic instance:

```
Enter the name of the WebLogic Server instance secured by the agent.
[ ? : Help, < : Back, ! : Exit ]
Enter the WebLogic Server instance name [AdminServer]: AdminServer
```

7. Review a summary of your responses and select how to continue:

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

Deploy Java Agent with Docker

The example in this section provides a Dockerfile and instructions to deploy Tomcat Java Agent on Linux to extend and protect an application. Adapt the information for other agent containers and platforms.

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 remote configuration mode. For more information, refer to Location of Agent Configuration Repository.

1. In PingOne Advanced Identity Cloud or AM, set up an agent profile and policy. For more information, refer to PingOne Advanced 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:8080/app`
   - Agent profile name: `java-agent`
   - Agent profile password: `password`
   - Policy set and policy: Allow HTTP `GET` and `POST` for all authenticated users.

2. Create a local folder for your application's `web.xml` file, the agent .zip file, the Dockerfile, and the agent profile password—they must be in the same folder. This example uses `/path/to/docker`.

3. Build a Docker image of your web application. This example uses a sample application called `fr-sample-app:1.0`.

4. Configure the agent filter in your application's `web.xml` file and save it to `/path/to/docker/web.xml`. For more information, refer to Configure the agent filter for a web application.

   This example uses the following `web.xml` file:

```
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN" "http://java.sun.com/dtd/
web-app_2_3.dtd">
<web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-
app_3_0.xsd">
 <filter>
  <filter-name>Agent</filter-name>
  <display-name>AM Agent</display-name>
  <description>AM Agent Filter</description>
  <filter-class>com.sun.identity.agents.filter.AmAgentFilter</filter-class>
 </filter>
 <filter-mapping>
  <filter-name>Agent</filter-name>
  <url-pattern>/*</url-pattern>
  <dispatcher>REQUEST</dispatcher>
  <dispatcher>INCLUDE</dispatcher>
  <dispatcher>FORWARD</dispatcher>
  <dispatcher>ERROR</dispatcher>
 </filter-mapping>
 <servlet>
  <servlet-name>ServletInfo</servlet-name>
  <servlet-class>org.forgerock.ServletInfo</servlet-class>
 </servlet>
</web-app>
```

5. Download the agent .zip file to `/path/to/docker/`. The .zip in this example is `tomcat_agent_${jpa.vers.ext}.zip`.

6. 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 using a password manager or by using the command line tool agentadmin --key.

7. 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=tomcat_agent_2024.3.zip
ARG APP_NAME
ARG AGENT_HOME=/opt
ARG AM_URL
ARG SERVER_HOME=/usr/local/tomcat
ARG AGENT_URL=http://agent.dummy.url:8080/app
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}

# Create an agent installation file called install_file
RUN echo "CONFIG_DIR= ${SERVER_HOME}/conf" > ${AGENT_HOME}/install_file && \
        echo "AM_SERVER_URL= ${AM_URL}" >> ${AGENT_HOME}/install_file && \
        echo "CATALINA_HOME= ${SERVER_HOME}" >> ${AGENT_HOME}/install_file && \
        echo "INSTALL_GLOBAL_WEB_XML= false" >> ${AGENT_HOME}/install_file && \
        echo "AGENT_URL= ${AGENT_URL}" >> ${AGENT_HOME}/install_file && \
        echo "AGENT_PROFILE_NAME= ${AGENT_PROFILE}" >> ${AGENT_HOME}/install_file && \
        echo "AGENT_PROFILE_REALM= ${AGENT_REALM}" >> ${AGENT_HOME}/install_file && \
        echo "AGENT_PASSWORD_FILE= /run/secrets/agent_secret" >> ${AGENT_HOME}/install_file

# Install the agent and mount the file containing the agent password
# This command uses silent installation with a provided install_file
RUN --mount=type=secret,id=agent_secret,required=true \
        "${AGENT_HOME}"/java_agents/tomcat_agent/bin/agentadmin \
        --forceInstall \
        --useResponse ${AGENT_HOME}/install_file && \
        rm -rf ${AGENT_HOME}/install_file

# Copy the new web.xml file, which includes agent filter
COPY web.xml ${AGENT_HOME}/

# Replace the original web.xml with the new web.xml file, which includes agent filter
RUN mkdir /tmp/app && \
        cd /tmp/app/ && \
        mv ${SERVER_HOME}/webapps/${APP_NAME} ./ && \
        jar -xf ./${APP_NAME} && \
        rm -rf ./${APP_NAME} && \
        mv ${AGENT_HOME}/web.xml ./WEB-INF/web.xml && \
        jar -cf ${SERVER_HOME}/webapps/${APP_NAME} * && rm -rf /tmp/app
```

8. Find values for the following arguments that correspond to your application and environment:
   - `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`. Define this property for Jakarta builds.
   - `APP_NAME`: Application name including the extension. For example, `app.war`
   - `AGENT_HOME`: Docker directory where the agent is installed. Default: `/opt`.
   - `AM_URL`: PingOne Advanced Identity Cloud or AM server URL including port number.
   - `SERVER_HOME`: Path to the Tomcat server configuration. Default: `/usr/local/tomcat`.
   - `AGENT_URL`: Agent URL.
   - `AGENT_REALM`: PingOne Advanced Identity Cloud or AM realm containing the agent profile.
   - `AGENT_PROFILE`: Agent profile name. Default `/`.
9. 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=${jpa.vers.ext} \
  --build-arg AGENT_ZIP_FILE=tomcat_agent_${jpa.vers.ext}.zip \
  --build-arg APP_NAME=app.war \
  --build-arg AGENT_HOME=/opt \
  --build-arg AM_URL=https://am.example.com:8443/am \
  --build-arg AGENT_URL=http://agent.example.com:8080/app \
  --build-arg SERVER_HOME=/usr/local/tomcat \
  --build-arg AGENT_REALM=/ \
  --build-arg AGENT_PROFILE=java-agent \
  --tag agent-image:${jpa.vers.ext} .

...
 => => writing image sha256:803...ada  0.0s
 => => naming to docker.io/library/java-agent:2023.11
```

10. Run the container:

```
/path/to/docker$ docker run -it --name java-agent -p 8080:8080 agent-image:2023.11

...INFO [main] org.apache.coyote.AbstractProtocol.start Starting ProtocolHandler...
...INFO [main] org.apache.catalina.startup.Catalina.start Server startup ...
```

11. Access your application through the agent at http://agent.example.com:8080/app. Access is managed by PingOne Advanced Identity Cloud or AM according to the policy configured for the agent profile.

This example displays the PingOne Advanced Identity Cloud or AM login 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 that corresponds to the version.
2. Replace the image tag in your environment.

agentadmin command

The `agentadmin` command manages Java Agent installation. It requires a Java runtime environment. The command supports the following options:

`--install`

Installs a new agent instance.

Usage: `agentadmin --install [--useResponse | --saveResponse file-name]`

Before installation, shut down the agent container. If a service on an agent URL is responding, the installer stops with an error.

When the command is used without options, the installation process prompts for the following information:

- Information about the container installation.
- The URL of the AM instance. The agent confirms that it can log in to AM by using the profile name and password provided during installation. If unsuccessful, the installation stops with an error.
- The URL of the agent instance. The agent confirms that it can access the host and port of the URL. If the port is busy, it prompts the user to stop the container.
- The agent profile name in AM.
- The AM realm containing the agent profile.
- The path to the file containing the agent password.

`--useResponse`

Run in silent mode by specifying all the responses in the file-name file. When this option is used, `agentadmin` runs in non-interactive mode.

`--saveResponse`

Save all the supplied responses in a response file specified by file-name.

`--forceInstall`

Installs a new agent instance, without checking the AM URL or agent URL.

Use this option in deployments with load balancers or reverse proxies, where the URL of the agent and AM can be concealed.

Usage: `agentadmin --forceInstall [--useResponse | --saveResponse file-name]`

Before installation, shut down the agent container. If a service on an agent URL is responding, the installer stops with an error.

When the command is used without options, the installation process prompts for the following information:

- Information about the container installation.
- The URL of the AM instance. The agent confirms that it can log in to AM by using the profile name and password provided during installation. If unsuccessful, the installation stops with an error.
- The URL of the agent instance. The agent confirms that it can access the host and port of the URL. If the port is busy, it prompts the user to stop the container.
- The agent profile name in AM.
- The AM realm containing the agent profile.
- The path to the file containing the agent password.

`--useResponse`

Run in silent mode by specifying all the responses in the file-name file. When this option is used, `agentadmin` runs in non-interactive mode.

`--saveResponse`

Save all the supplied responses in a response file specified by file-name.

`--custom-install`, `--custom`

Installs a new agent instance, specifying advanced configuration options.

Usage: `agentadmin --custom-install [--useResponse | --saveResponse file-name]`

`--useResponse`

Run in silent mode by specifying all the responses in the file-name file. When this option is used, `agentadmin` runs in non-interactive mode.

`--saveResponse`

Save all the supplied responses in a response file specified by file-name.

`--uninstall`, `-r`

Uninstalls an existing agent instance.

Usage: `agentadmin --uninstall [--useResponse | --saveResponse file-name]`

`--useResponse`

Run in silent mode by specifying all the responses in the file-name file. When this option is used, `agentadmin` runs in non-interactive mode.

`--saveResponse`

Save all the supplied responses in a response file specified by file-name.

`--version`, `-v`

Displays the agent version.

Usage: `agentadmin --version`

`--uninstallAll`

Uninstalls all agent instances.

Usage: `agentadmin --uninstallAll`

`--listAgents`, `--list`, `-l`

Displays information about all configured agents.

Usage: `agentadmin --listAgents`

`--agentInfo`, `--info`

Displays information about the agent corresponding to the specified agent-id.

Usage: `agentadmin --agentInfo agent-id`

Example: `agentadmin --agentInfo agent_001`

`--encrypt`

Encrypts a given string.

Usage: `agentadmin --encrypt agent-instance password-file`

agent-instance

Agent instance identifier. The encryption functionality requires the use of agent instance specific encryption key present in its configuration file.

password-file

File containing the password to encrypt.

`--getEncryptKey, --getKey`

Generates an agent encryption key of 40 characters long.

Usage: `agentadmin --getEncryptKey`

`--key`

Generates an agent encryption key of the specified length. For security, generate keys that are about 80 characters long.

Usage: `agentadmin --key key-length`

`--d, -d, --decryptAgent, --decrypt`

Reveals the agent password in clear text, for the agent corresponding to the specified agent-id.

Usage: `agentadmin --d [agent-id]`

Example: `agentadmin --d Agent_001`

agent-id

The agent instance. Default: `Agent_001`.

`--decryptPassword`

Decrypts the agent password, for the agent corresponding to the specified agent-id.

Usage: `agentadmin --decryptPassword encrypted-password encryption-key`

encrypted-password

Encrypted agent password.

encryption-key

Key used to encrypt the agent password.

Upgrade

Java Agent supports the following types of upgrade:

- Drop-in software update:

  Usually, an update from a version of Java Agent 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 Java Agent to a newer major version, as defined in Ping Identity Product Support Lifecycle Policy | PingGateway and Agents. For example, upgrade from 2023.3 to 2024.3 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 PingAM Java Agent instance. To upgrade sites with multiple Java Agent 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 Java Agent, 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:

- Java Agent installed on `http://agent.example.com:80`.
- PingAM installed on `http://am.example.com:8080/am`.
- Work in the top-level realm `/`.

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

Drop-in software update

The examples in this section assume that the agent is installed in `/path/to/java_agents/agent_type`, and the update is from the minor version ${previous.minor.version} to the minor version ${current.minor.version}.

Tomcat Java Agent software update

1. Read the release notes for information about changes in Java Agent.
2. Download the agent binaries from the Backstage download site and extract them to a temporary directory.

   The example in this section is extracted to `/tmp`, and the .jar files are in `/tmp/tomcat_agent/lib`.
3. Back up the directories for the agent installation and the web application container configuration:
   - In local configuration mode:

     ```
     $ cp -r /path/to/java_agents/tomcat_agent /path/to/backup
     $ cp -r /path/to/tomcat/webapps/agentapp /path/to/backup
     ```
   - In remote configuration mode, back up as described in AM's Maintenance guide.
4. Redirect client traffic away from protected web applications.
5. Stop the web applications where the agent is installed.
6. Locate the following files in the container:
   - `agent.jar`
   - `jee-agents-sdk-version.jar`

   The following example finds `./lib/jee-agents-sdk-${previous.minor.version}.jar`:

Unix

```
$ cd /opt/container
$ find . -type f -name 'jee-agents-*.jar' -print
./lib/jee-agents-sdk-${previous.minor.version}.jar
```

Windows

```
C:\> cd C:\opt\container
C:\opt\container> dir /s jee-agents-*.jar

...
  Directory of C:\opt\container\lib

  date  time          ... jee-agents-sdk-${previous.minor.version}.jar
...
```

7. If `agent.jar` is present in the container, delete it.
8. Replace `jee-agents-sdk-version.jar` with the newer downloaded version. The following example replaces `jee-agents-sdk-${previous.minor.version}.jar`:

Unix

```
$ cd /opt/container
$ rm -f lib/jee-agents-sdk-${previous.minor.version}.jar
$ cp /tmp/tomcat_agent/lib/jee-agents-sdk-${current.minor.version}.jar lib
```

Windows

```
C:\opt\container> del lib\jee-agents-sdk-${previous.minor.version}.jar
C:\opt\container> copy C:\tmp\tomcat_agent\lib\jee-agents-sdk-${current.minor.version}.jar lib
```

9. (Optional) Update the .jar files outside the container.
    1. Using the `.amAgentLocator` file, find the directory in which the agent was originally installed:

Unix

```
$ cd /opt/container
$ cat .amAgentLocator; echo

/path/to/java_agents/tomcat_agent
```

Windows

```
C:\opt\container> type .amAgentLocator

C:/path/to/java_agents/tomcat_agent
```

2. View the content of the `lib` subdirectory:

Unix

```
$ cd /path/to/java_agents/tomcat_agent/lib
$ ls -F

agent.jar
jee-agents-installtools-${previous.minor.version}.jar
jee-agents-sdk-${previous.minor.version}.jar
```

Windows

```
C:\opt\container> cd C:\path\to\java_agents\tomcat_agent\lib
C:\path\to\java_agents\tomcat_agent\lib> dir

Directory of C:\path\to\java_agents\tomcat_agent\lib
... agent.jar
... jee-agents-installtools-${previous.minor.version}.jar
... jee-agents-sdk-${previous.minor.version}.jar
```

3. Replace the files with the newer downloaded version:

Unix

```
$ rm -f *
$ cp /tmp/java_agents/tomcat_agent/lib/*.jar .
$ ls -F

agent.jar
jee-agents-installtools-${current.minor.version}.jar
jee-agents-sdk-${current.minor.version}.jar
```

Windows

```
C:\path\to\java_agents\tomcat_agent\lib> del *.jar
C:\path\to\java_agents\tomcat_agent\lib> copy C:
\temp\java_agents\tomcat_agent\lib\*.jar .

C:\tmp\tomcat_agent\lib\agent.jar
C:\tmp\tomcat_agent\lib\jee-agents-installtools-${current.minor.version}.jar
C:\tmp\tomcat_agent\lib\jee-agents-sdk-${current.minor.version}.jar
```

10. Replace the current `agentadmin` file with the newer downloaded version:

Unix

```
$ cd /path/to/java_agents/tomcat_agent/bin
$ rm agentadmin
$ cp /tmp/tomcat_agent/bin/agentadmin .
```

Windows

```
C:\> cd C:\path\to\java_agents\tomcat_agent\bin
C:\path\to\java_agents\tomcat_agent\bin> del agentadmin agentadmin.bat
C:\path\to\java_agents\tomcat_agent\bin> copy C:\tmp\tomcat_agent\bin\agentadmin .
C:\path\to\java_agents\tomcat_agent\bin> copy C:\tmp\tomcat_agent\bin\agentadmin.bat .
```

11. Start the web applications where the agent is installed.
12. Check that the agent is performing as expected:
    1. Check the correct version of the agent is running:
       - Set the log level to `trace`, as described in Logging.
       - In `/path/to/java_agents/agent_type/Agent_n/logs/debug`, search for lines containing the string `X-ForgeRock-Edge-Metadata`. The version number is given in the header.

         For example, the log file can contain the following header: `--header "X-ForgeRock-Edge-Metadata: JPA ${current.minor.version}`.
    2. Navigate to a protected page on the website and confirm whether you can access it according to your configuration.
    3. Check logs files for warnings and errors.

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

JBoss and WildFly Java Agent software update

1. Read the release notes for information about changes in Java Agent.
2. Download the agent binaries from the Backstage download site and extract them to a temporary directory.

   The example in this section is extracted to `/tmp`, and the .jar files are in `/tmp/jboss_agent/lib`.
3. Back up the directories for the agent installation and the web application container configuration:
   - In local configuration mode:

     ```
     $ cp -r /path/to/java_agents/jboss_agent /path/to/backup
     $ cp -r /path/to/jboss/webapps/agentapp /path/to/backup
     ```

   - In remote configuration mode, back up as described in AM's Maintenance guide.
4. Redirect client traffic away from protected web applications.
5. Stop the web applications where the agent is installed.
6. Update the `module.xml` file.
   1. Locate the path to the installation, for example, at `/path/to/jboss/modules/org/forgerock/openam/agent/main/modules/org/forgerock/openam/agent/main`.
   2. If any of the following files are listed, remove the resource for the file:
      - `tyrus-standalone-client-2.1.3.jar`
      - `jee-agents-jboss-common-${previous.minor.version}.jar`
      - `agent.jar`
   3. Update the resource for `jee-agents-sdk-version.jar` to use the absolute path and the newer downloaded version agent version. For example, change

      ```
      <resource-root path="jee-agents-sdk-${previous.minor.version}.jar"/>
      ```

      to

      ```
      <resource-root path="/path/to/java_agents/jboss_agent/lib/jee-agents-sdk-$
      {current.minor.version}.jar"/>
      ```
7. Update the .jar files outside the container.
   1. Using the `.amAgentLocator` file, find the directory in which the agent was originally installed:

```
$ cd /opt/container
$ cat .amAgentLocator; echo

/path/to/java_agents/jboss_agent
```

2. View the content of the `lib` subdirectory:

```
$ cd /path/to/java_agents/jboss_agent/lib
$ ls -F

agent.jar
jee-agents-jboss-common-version.jar
jee-agents-sdk-version.jar
tyrus-standalone-client-version.jar
```

3. Replace the files with the newer downloaded version:

```
$ rm -f *
$ cp /tmp/java_agents/jboss_agent/lib/*.jar .
$ ls -F

agent.jar
jee-agents-jboss-common-version.jar
jee-agents-sdk-version.jar
tyrus-standalone-client-version.jar
```

8. Replace the current `agentadmin` file with the newer downloaded version:

```
$ cd /path/to/java_agents/jboss_agent/bin
$ rm agentadmin
$ cp /tmp/jboss_agent/bin/agentadmin .
```

9. Start the web applications where the agent is installed.
10. Check that the agent is performing as expected:
    1. Check the correct version of the agent is running:
        - Set the log level to `trace`, as described in Logging.
        - In `/path/to/java_agents/agent_type/Agent_n/logs/debug`, search for lines containing the string `X-ForgeRock-Edge-Metadata`. The version number is given in the header.

          For example, the log file can contain the following header: `--header "X-ForgeRock-Edge-Metadata: JPA ${current.minor.version}`.
    2. Navigate to a protected page on the website and confirm whether you can access it according to your configuration.
    3. Check logs files for warnings and errors.

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

Jetty Java Agent software update

1. Read the release notes for information about changes in Java Agent.
2. Download the agent binaries from the Backstage download site and extract them to a temporary directory.

   The example in this section is extracted to `/tmp`, and the .jar files are in `/tmp/jetty_agent/lib`.
3. Back up the directories for the agent installation and the web application container configuration:
   - In local configuration mode:

     ```
     $ cp -r /path/to/java_agents/jetty_agent /path/to/backup
     $ cp -r /path/to/jetty/webapps/agentapp /path/to/backup
     ```
   - In remote configuration mode, back up as described in AM's Maintenance guide.
4. Redirect client traffic away from protected web applications.
5. Stop the web applications where the agent is installed.
6. Replace the following files with the newer downloaded versions.
   - `agent.jar`
   - `jee-agents-installtools-version.jar`
   - `jee-agents-sdk-version.jar`

   The following example replaces `jee-agents-sdk-${previous.minor.version}.jar`:

   ```
   $ cd /path/to/java_agents/jetty_agent/lib
   $ rm -f jee-agents-sdk-${previous.minor.version}.jar
   $ cp /tmp/jetty_agent/lib/jee-agents-sdk-${current.minor.version}.jar .
   ```
7. Update the Jetty configuration:
   1. Go to the Jetty base directory.

      ```
      $ cd /path/to/jetty-base/modules
      ```
   2. In `amlogin.mod`, delete the line for `/path/to/java_agents/jetty_agent/lib/agent.jar` if it is present. This file isn't required from Java Agent 2023.9.
   3. In `amlogin.mod`, update the version number for `jee-agents-sdk-version.jar`. The following example includes `jee-agents-sdk-${current.minor.version}.jar`:

```
# Jetty AM module
#
[depend]
server
security
jndi
webapp
plus
[xml]
etc/amlogin.xml
[lib]
/path/to/java_agents/jetty_agent/Agent_001/config
/path/to/java_agents/jetty_agent/locale
/path/to/java_agents/jetty_agent/lib/jee-agents-sdk-${previous.minor.version}.jar
```

8. Replace the current `agentadmin` file with the newer downloaded version:

```
$ cd /path/to/java_agents/jetty_agent/bin
$ rm agentadmin
$ cp /tmp/jetty_agent/bin/agentadmin .
```

9. Start the web applications where the agent is installed.
10. Check that the agent is performing as expected:
    1. Check the correct version of the agent is running:
        - Set the log level to `trace`, as described in Logging.
        - In `/path/to/java_agents/agent_type/Agent_n/logs/debug`, search for lines containing the string `X-ForgeRock-Edge-Metadata`. The version number is given in the header.

          For example, the log file can contain the following header: `--header "X-ForgeRock-Edge-Metadata: JPA ${current.minor.version}`.
    2. Navigate to a protected page on the website and confirm whether you can access it according to your configuration.
    3. Check logs files for warnings and errors.
11. Allow client traffic to flow to the protected web applications.

WebLogic Java Agent software update

1. Read the release notes for information about changes in Java Agent.
2. Download the agent binaries from the Backstage download site and extract them to a temporary directory.

   The example in this section is extracted to `/tmp`, and the .jar files are in `/tmp/weblogic_agent/lib`.
3. Back up the directories for the agent installation and the web application container configuration:
    - In local configuration mode:

```
$ cp -r /path/to/java_agents/weblogic_agent /path/to/backup
```

- In remote configuration mode, back up as described in AM's Maintenance guide.

4. Add the following file to the backup:
   - `/Oracle/Middleware/Oracle_Home/user_projects/domains/base_domain/setAgentEnv_AdminServer.sh`
5. Redirect client traffic away from protected web applications.
6. Stop the web applications where the agent is installed.
7. Update the .jar files in the installation directory.
   1. Using the `.amAgentLocator` file, find the directory in which the agent was originally installed:

```
$ cd /opt/container
$ cat .amAgentLocator; echo

/path/to/java_agents/weblogic_agent
```

   2. View the content of the `lib` subdirectory:

```
$ cd /path/to/java_agents/weblogic_agent/lib
$ ls -F

agent.jar
jee-agents-installtools-${previous.minor.version}.jar
jee-agents-sdk-${previous.minor.version}.jar
```

   3. Replace the files with the newer downloaded version:

```
$ rm -f *
$ cp /tmp/java_agents/weblogic_agent/lib/*.jar .
$ ls -F

agent.jar
jee-agents-installtools-${current.minor.version}.jar
jee-agents-sdk-${current.minor.version}.jar
```

8. Update the environment settings:
   1. Locate the `setAgentEnv_AdminServer.sh` file. The file can be in a directory such as `/Oracle/Middleware/Oracle_Home/user_projects/domains/base_domain/`.
   2. If any of the following files are listed, remove the information for the file:
      - `/path/to/java_agents/weblogic_agent/lib/agent.jar`.

- `/path/to/java_agents/weblogic_agent/lib/jee-agents-installtools-launcher-version`. The installation launcher was removed in Java Agent 2023.6.
- `/path/to/java_agents/weblogic_agent/lib/jee-agents-installtools-version.jar`.

3. Change the version of `jee-agents-sdk-version.jar` to the newer downloaded version:

```
...
# Append AGENT_CLASSPATH to the WebLogic server classpath
AGENT_CLASSPATH="/path/to/java_agents/weblogic_agent/lib/jee-agents-sdk-${current.minor.version}.jar:/
path/to/java_agents/weblogic_agent/locale:/path/to/java_agents/weblogic_agent/Agent_001/config"
CLASSPATH="${CLASSPATH}${CLASSPATHSEP}${AGENT_CLASSPATH}"
export CLASSPATH
...
```

4. Save the file.

9. Replace the current `agentadmin` file with the newer downloaded version:

```
$ cd /path/to/java_agents/weblogic_agent/bin
$ rm agentadmin
$ cp /tmp/weblogic_agent/bin/agentadmin .
```

10. Start the web applications where the agent is installed.
11. Check that the agent is performing as expected:
    1. Check the correct version of the agent is running:
       - Set the log level to `trace`, as described in Logging.
       - In `/path/to/java_agents/agent_type/Agent_n/logs/debug`, search for lines containing the string `X-ForgeRock-Edge-Metadata`. The version number is given in the header.

         For example, the log file can contain the following header: `--header "X-ForgeRock-Edge-Metadata: JPA ${current.minor.version}`.
    2. Navigate to a protected page on the website and confirm whether you can access it according to your configuration.
    3. Check logs files for warnings and errors.
12. Allow client traffic to flow to the protected web applications.

Roll back from a drop-in software update

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

Major upgrade

Perform a major upgrade

1. Read the release notes for information about changes in Java Agent.
2. 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 applications are aware of the change, and let them know what to expect.
3. Download the agent binaries from the Backstage download site.
4. Back up the directories for the agent installation and the web application container configuration:
   - In local configuration mode:

     ```
     $ cp -r /path/to/java_agents/agent_type /path/to/backup
     $ cp -r /path/to/agent_type/webapps/agentapp /path/to/backup
     ```
   - In remote configuration mode, back up as described in AM's Maintenance guide.
5. Redirect client traffic away from protected web applications.
6. Stop the web applications where the agent is installed.
7. Remove the old Java Agent, as described in Remove Java Agent.
8. Install the new agent.

   The installer creates new versions of the following files, with configuration that is relevant to the new version of the agent:
   - AgentConfiguration.properties
   - AgentBootstrap.properties
   - agent-logback.xml
   - AgentPassword.properties
   - AgentKey.properties
9. Using the agent's release notes and AM's release notes, check for changes and update the configuration.

   Important

   To prevent errors, do not copy configuration files from the previous installation to the new installation. Use the new version of the files and update then as necessary.
   - In local configuration mode, update `AgentConfiguration.properties` manually to include properties for your environment, using backed-up files for guidance.

The `AgentBootstrap.properties` file created by the installer contains bootstrap properties relevant to the new version of the agent.

- In remote configuration mode, change the agent configuration using the AM admin UI.

10. Secure communication between AM and the agent with appropriate keys. For information, refer to Configure AM servers to communicate with Java Agents.
11. Start the web applications where the agent is installed.
12. Check that the agent is performing as expected:
    1. Check the correct version of the agent is running:
       - Set the log level to `trace`, as described in Logging.
       - In `/path/to/java_agents/agent_type/Agent_n/logs/debug`, search for lines containing the string `X-ForgeRock-Edge-Metadata`. The version number is given in the header.

         For example, the log file can contain the following header: `--header "X-ForgeRock-Edge-Metadata: JPA ${current.minor.version}`.
    2. Navigate to a protected page on the website and confirm whether you can access it according to your configuration.
    3. Check logs files for warnings and errors.
13. Allow client traffic to flow to the protected web applications.

Roll back from a major upgrade

> **Important**
>
> Before you roll back to a previous version of Java Agent, consider whether any change to the configuration during or since upgrade could be incompatible with the previous version.

Post update and upgrade tasks

After upgrade or update, 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 PingAM Java Agent.

About Ping Identity Platform™ software

Ping 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 Java Agent

Java Agent is an PingAM add-on component that operates as a Policy Enforcement Point (PEP) or policy agent for web applications deployed on a Java container.

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

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

This chapter covers how Java Agents work and how their features can protect web applications.

Agent components

Java Agent includes the following main components:

Agent Filter

Intercepts inbound client requests to a resource and processes them based on the filter mode of operation.

Agent Application

Deployed as `agentapp.war`, it is required for authentication and the cross-domain single sign-on (CDSSO) flow.

The following components are not part of Java Agent, but they play an important part in the agent operation:

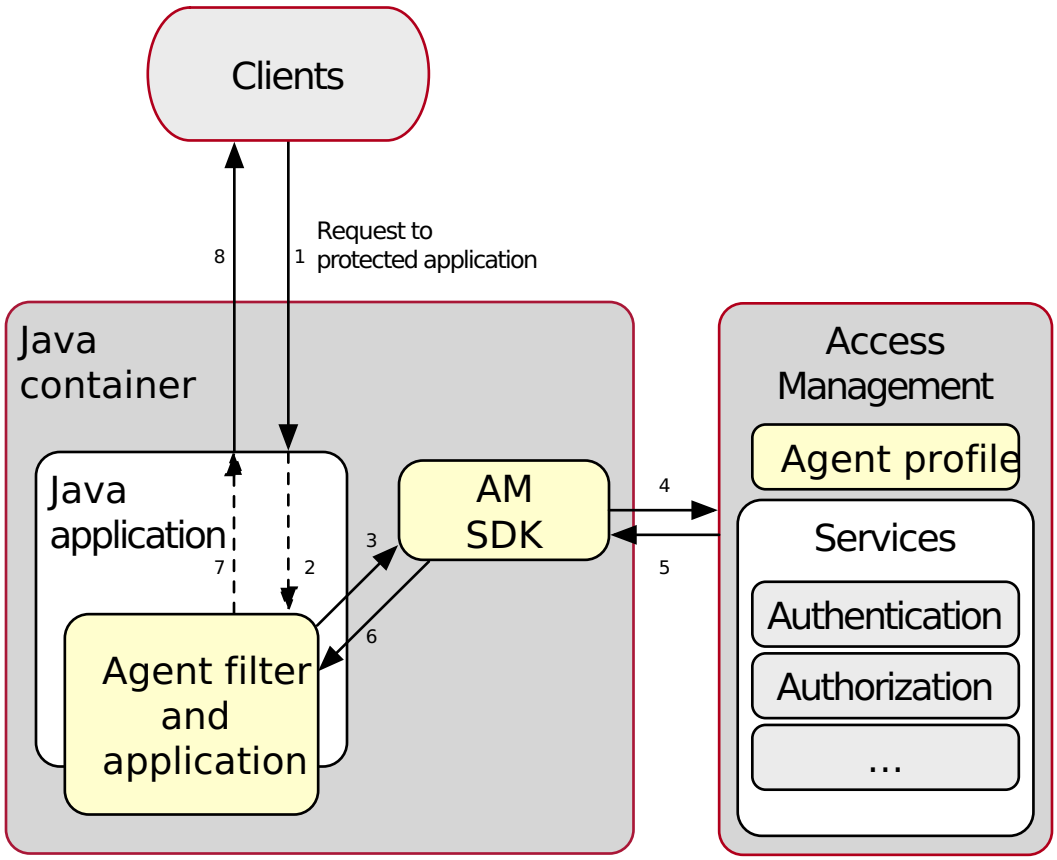
AM SDKs

A set of APIs required to interact with AM.

Agent Profile

A set of configuration properties that define the agent behavior. The agent profile can be stored in AM's configuration store or as a text file local to the agent installation.

The following image shows the Java Agent components when the agent profile is stored in the AM configuration store:



Agent configuration

Java Agent uses the configuration files described in this section.

The files must be in a location defined by a property added to `JAVA_OPTS`. For example, in Tomcat, the agent can take the file location from `bin/setenv.sh`, as follows:

```
JAVA_OPTS="$JAVA_OPTS -Dopenam.agents.bootstrap.dir=/path/to/agents/agent/agent_instance/config"
```

AgentBootstrap.properties

This file defines bootstrap parameters. The following information is required in the file:

- Private AM URL:

  Used for communication with AM, for example, to retrieve policy information or user information. The URL is assembled from the following properties, and is required, even if the agent never contacts AM:

  - AM Authentication Service Protocol
  - AM Authentication Service Host Name
  - AM Authentication Service Port
  - AM Authentication Service Path

- Public AM URL:

This URL must be provided by the user if the AM firewall rules distinguish between a public and a private URL. The agent uses this property to redirect the user's browser to public-facing URLs for login. If it is not provided, the AM private URL is used.

- Agent Profile:
  - Agent Profile Name
  - Agent Profile Realm
- Location of Agent Configuration Repository:

  Defines the agent configuration mode:

  - Local configuration mode

    The agent reads its configuration from `AgentConfiguration.properties`. When the agent is in this mode, it ignores changes made to the agent profile in AM.

    Depending on the configuration in the `AgentConfiguration.properties` file, the agent might never need to contact AM. For example, if Autonomous mode is `true`, the agent runs independently of an AM instance.

  - Remote configuration mode (default)

    The agent ignores the configuration in `AgentConfiguration.properties`, retains the retrieved bootstrap properties, and downloads the configuration from AM.

    When the first user request is made, the agent contacts AM to retrieve the agent configuration. If AM is unavailable, the request returns an `HTTP 503 Service Unavailable`.

AgentConfiguration.properties

This file defines agent configuration settings, and overrides any properties set in the bootstrap file.

If the value of Location of Agent Configuration Repository is `LOCAL`, the agent reads the `AgentConfiguration.properties` file after `AgentBootstrap.properties`. If the value of Location of Agent Configuration Repository is `REMOTE`, the agent ignores this file.

In Java Agent 5.7 and earlier versions, this file was named `OpenSSOAgentConfiguration.properties`.

AgentPassword.properties

This file defines an encrypted password for the agent profile. For more information, refer to Encrypted Agent Password.

AgentKey.properties

This file defines the following keys:

- The encryption key for the agent profile password. For more information, refer to Encryption Key/Salt.
- The signing key for pre-authentication cookies and POST data preservation cookies. For more information, refer to Pre-Authn and Post Data Preservation Cookie Signing Value.

agent-logback.xml

This file configures logging of Java Agent and third-party dependencies, using the Logback implementation of the Simple Logging Facade for Java (SLF4J) API. For more information, refer to Logging.

Changing the agent configuration

Change the agent configuration in the following ways:

Change the agent bootstrap configuration

Manually edit `AgentBootstrap.properties`, and then restart the container running the agent.

Change the agent configuration in LOCAL mode

Manually edit the `AgentConfiguration.properties` file, and set a value for Configuration Reload Interval.

The interval defines the number of seconds after which the agent reads the local property file, and reloads it if has changed since it was last read.

The value of Location of Agent Configuration Repository must be `LOCAL`.

Change the agent configuration in REMOTE mode

The agent is notified by the WebSocket mechanism when its configuration is changed in AM. The agent then re-reads its configuration from AM within a few seconds.

The value of Location of Agent Configuration Repository must be `REMOTE`.

Change the agent configuration on the AM admin UI

Go to **Realms** > realm name > **Application** > **Agents** > **Java** > agent name.

The value of Location of Agent Configuration Repository must be `REMOTE`.

Realms

Agent profile realm

The agent profile stores a set of configuration properties that define the behavior of the agent.

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

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

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 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 Map, and defaults to the top-level realm. The policy set to use is configured by Policy Set Map To ensure that policies are always evaluated in the user realm, set Enable Policy Evaluation in User Authentication Realm to `true`.

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 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, Java Agent uses the following properties to obtain a session from AM:

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

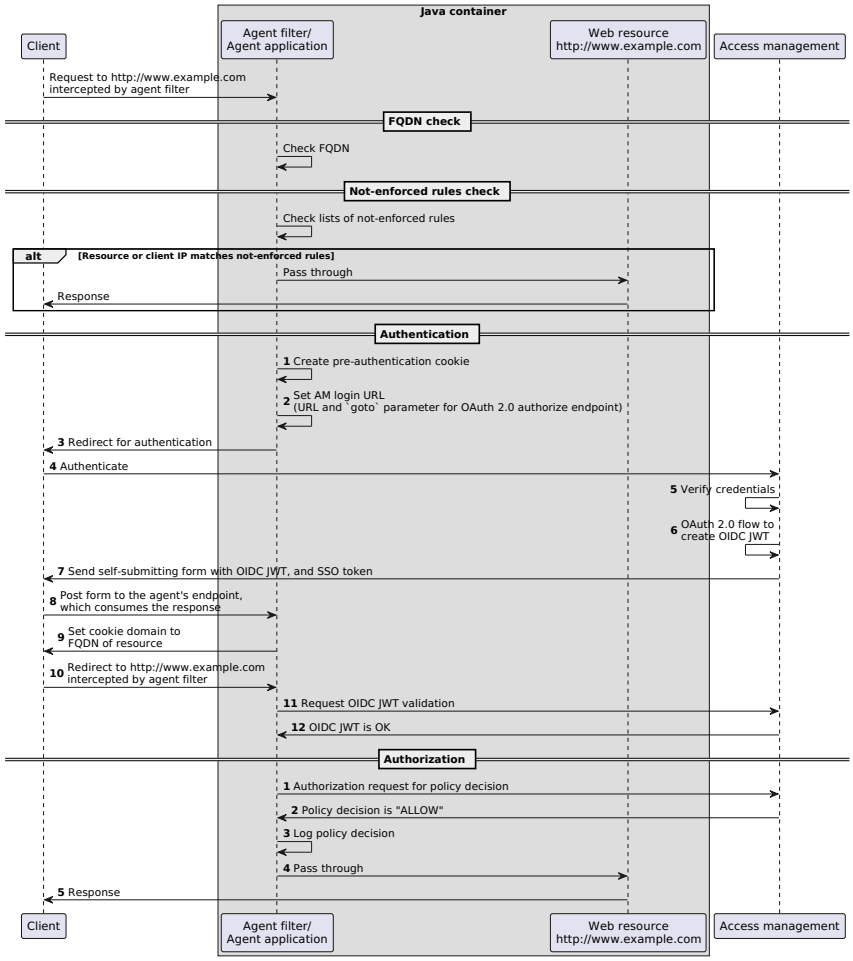
The agent session lifetime is defined by the AM version and configuration, and is essentially indefinite.

For the security of your deployment, set the agent session lifetime as described in Manage Java Agent 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.

Request flow

The following simplified data flow occurs when an unauthenticated client requests a resource protected by a Java Agent and AM. The flow assumes that requests must meet the requirements of an AM policy. For more information, see 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.

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

For more information, refer to Not-enforced rules.

Authentication

1: The agent creates a pre-authentication cookie to protect against reply attacks. The agent uses this cookie to track the authentication request to AM. Depending on the configuration, the agent may either issue a cookie to track all concurrent authentication requests, or may issue one cookie for each request. For added security, the pre-authentication cookie can be optionally be signed.

2: The agent sets the AM login URL, which includes the `goto` parameter for the OAuth 2.0 authorize endpoint, and

3: The agent redirects the client to log in to AM.

4-7: The client authenticates to AM:

- AM's Authentication Service verifies the client credentials. AM creates an SSO token, and OIDC JWT with session information.
- AM sends the client a self-submitting form with the OIDC JWT.

8: The client posts the self-submitting form to the agent endpoint, and the Java Agent consumes it.

9: The agent sets the cookie domain to the FQDN of the resource.

10: The client attempts to access the protected resource again, and the agent intercepts the request.

11: The agent contacts AM to validate the session contained in the OIDC JWT.

12: 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.

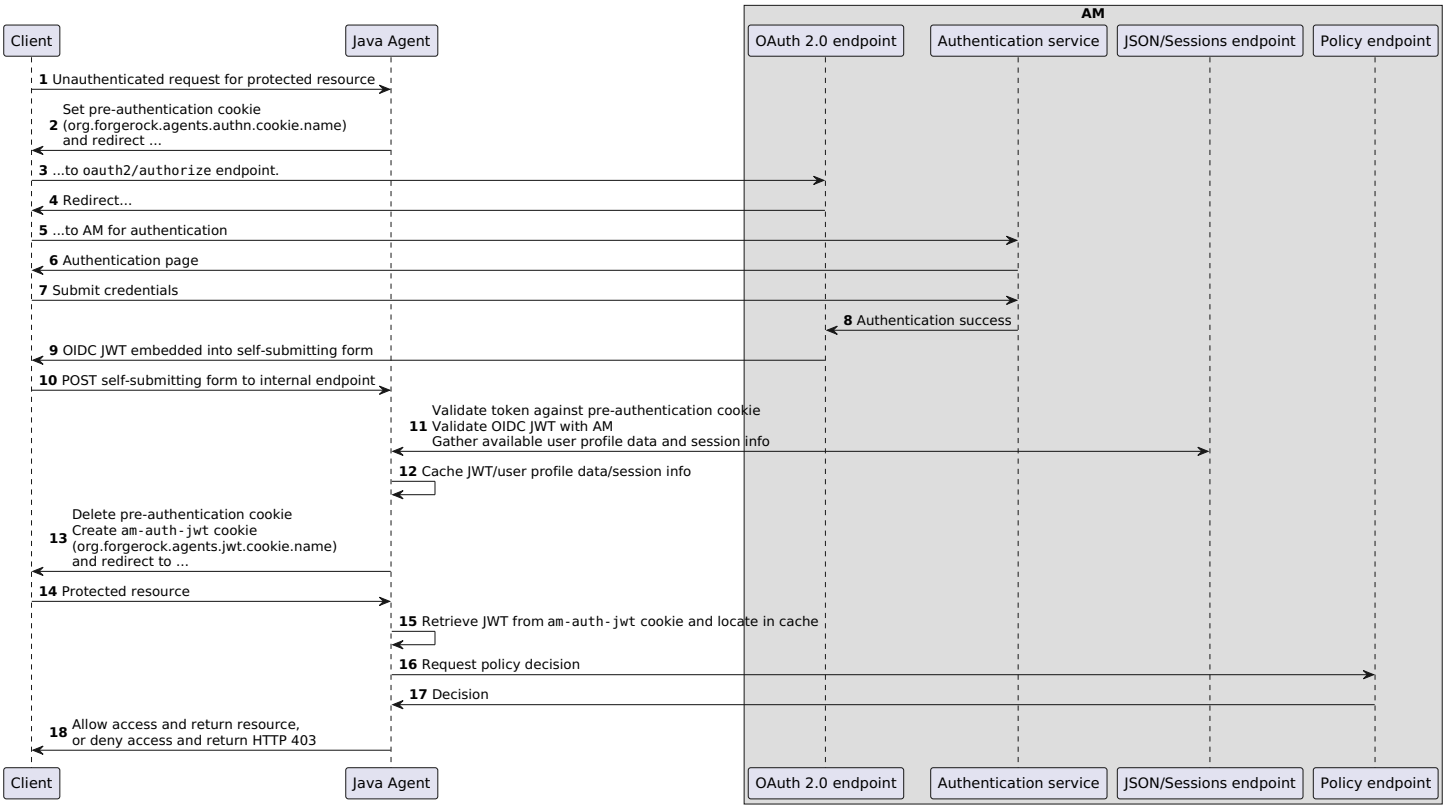
4: The agent enforces the policy decision. Because the Policy Service returned `ALLOW`, the agent performs a pass-through operation to return the resource to the client.

5: The client accesses the resource.

Cross-domain single sign-on

In Cross-Domain Single Sign-On (CDSSO), Java Agent processes requests using authentication provided by AM. Users can access multiple independent services from a single login session, using the agent to transfer the session ID. The agent and AM can be in the same domain or in different domains.

The following diagram illustrates the CDSSO flow:



When the agent is in local configuration mode, configure the Authentication Redirect URI. When the agent is in remote configuration mode, the value is set by the agent configuration in AM.

For more information, refer to Single sign-on and Implement CDSSO in AM's *Authentication and SSO guide*.

Policy enforcement

This example sets up AM as a policy decision point for requests processed by Java Agent.

Before you start, install a Java Agent as described in the Installation, with the following values:

- AM server URL: `http://am.example.com:8080/am`
- Agent URL: `http://agent.example.com:80`
- Agent profile name: `java-agent`
- Agent profile realm: `/`
- Agent profile password: `/secure-directory/pwd.txt`

Enforce a policy decision from AM

1. Using the PingAM docs for information, log in to AM as an administrator, and make sure that you are managing the `/` realm.

2. Add a Java Agent profile:
    1. In the AM admin UI, select **Applications** > **Agents** > **Java**.
    2. Add an agent with the following values:
        - Agent ID: `java-agent`
        - Agent URL: `http://agent.example.com:80`
        - Server URL: `http://am.example.com:8080/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** > **Java**, 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 Java Agent.

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 attribute 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*.
- 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, HTTP cookies, or request attributes 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 `java-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 select whether to map response attribute names to HTTP headers, HTTP cookies, or request attributes. 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 an HTTP header, HTTP cookie, or request attribute called `CUSTOM-name`. The value is taken from the user profile.

    For more information, refer to Response Attribute Map.

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 `Frequent flyer status` is mapped to an HTTP header, HTTP cookie, or request attribute called `CUSTOM-flyer-status`. The value is `gold`.

      For more information, refer to Response Attribute Map.

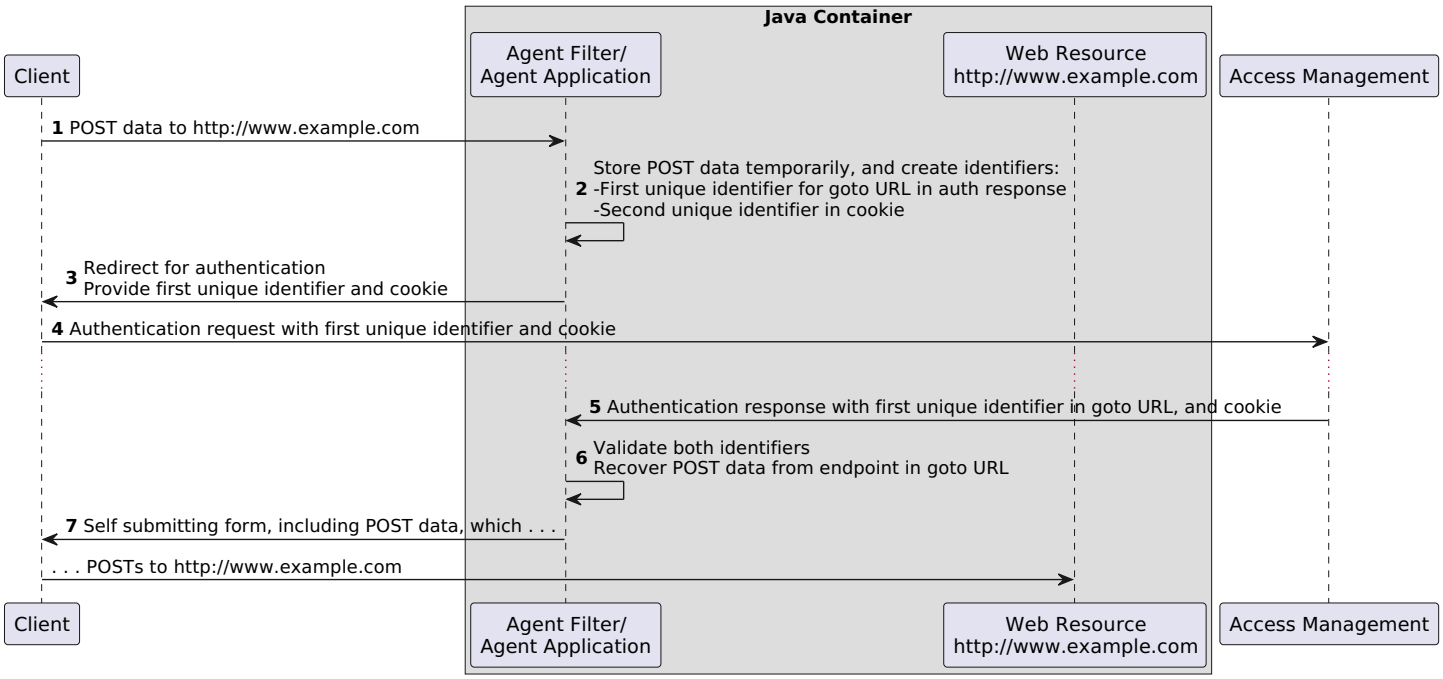
POST data preservation

When POST data preservation is enabled and an unauthenticated client POSTs data to a protected resource, the agent stores the data in the POST data preservation cache and redirects the client for login. After successful authentication, the agent recovers the cached data and automatically submits it to the protected resource.

The POST data can be any POST content, such as a single-part HTML form or a file upload.

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

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



Java Agent 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, and then generates the following unique identifiers:
    - An identifier in the goto URL for the authentication response
    - An identifier in a cookie

The use of two unique identifiers provides robust security, because a hacker must steal the goto URL and the cookie.

3. The agent redirects the client to AM for authentication, and includes the cookie in the redirect.
4. The client authenticates with AM.
5. AM provides an authentication response to the goto URL with the unique identifier, and includes the cookie.
6. The agent validates both identifiers, and recovers the POST data from the dummy internal endpoint given in the goto URL.

   If the goto URL contains the incorrect identifier, or cannot provide a cookie containing the correct second identifier (for example, because it has expired), the agent denies the request.

   The presence of the unique identifier in the goto URL ensures that requests at the URL can be individually identified. Additionally, the code makes it more difficult to hijack user data, because there is little chance of guessing the code within the login window.

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.

Configure POST data preservation

Configure POST data preservation by using the agent properties listed in POST Data Preservation in the *Properties reference*, or on the **Advanced** tab of the AM admin UI.

Security considerations for storing POST data in files

By default, POST data is stored in the in-memory cache. Consider the following points if you configure POST Data Preservation in Files or Cache to store POST data in the file system:

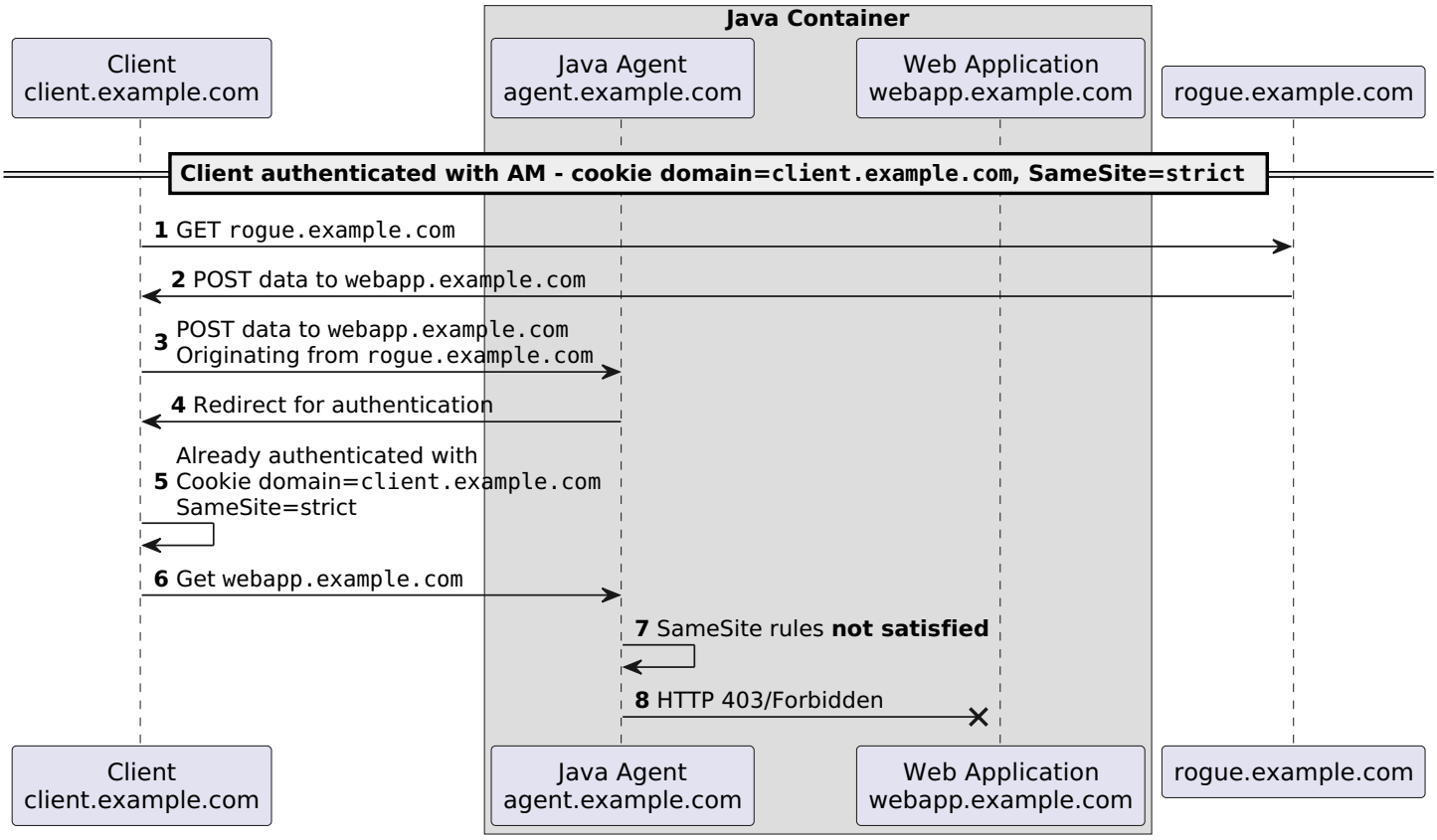
- Payloads from unauthenticated users are stored in the agent file system. If your threat evaluation does not accept this risk, store the data in the cache, or set POST Data Preservation in Files or Cache to `false`.
- Restrict access to the POST Data Preservation File Directory, to mitigate the risk of permissive access or leakage of personally identifiable information (PII).
- Limit the amount of stored POST data to mitigate the risk of DoS attacks, by configuring POST Data Preservation Storage Size or Max Entries in POST Data Preservation Storage.
- Remove expired POST data as soon as possible by configuring the POST Data Preservation Directory Sweep Interval.
- Identify threats in POST data before it is deleted, by making sure that Intrusion Detection Systems inspect the data within the time specified by POST Data Preservation Directory Sweep Interval.

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 Java Agent 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. ForgeRock strongly recommends using token-based mitigations against CSRF, and relying on other measures only as a defense in depth, in accordance with OWASP guidance.

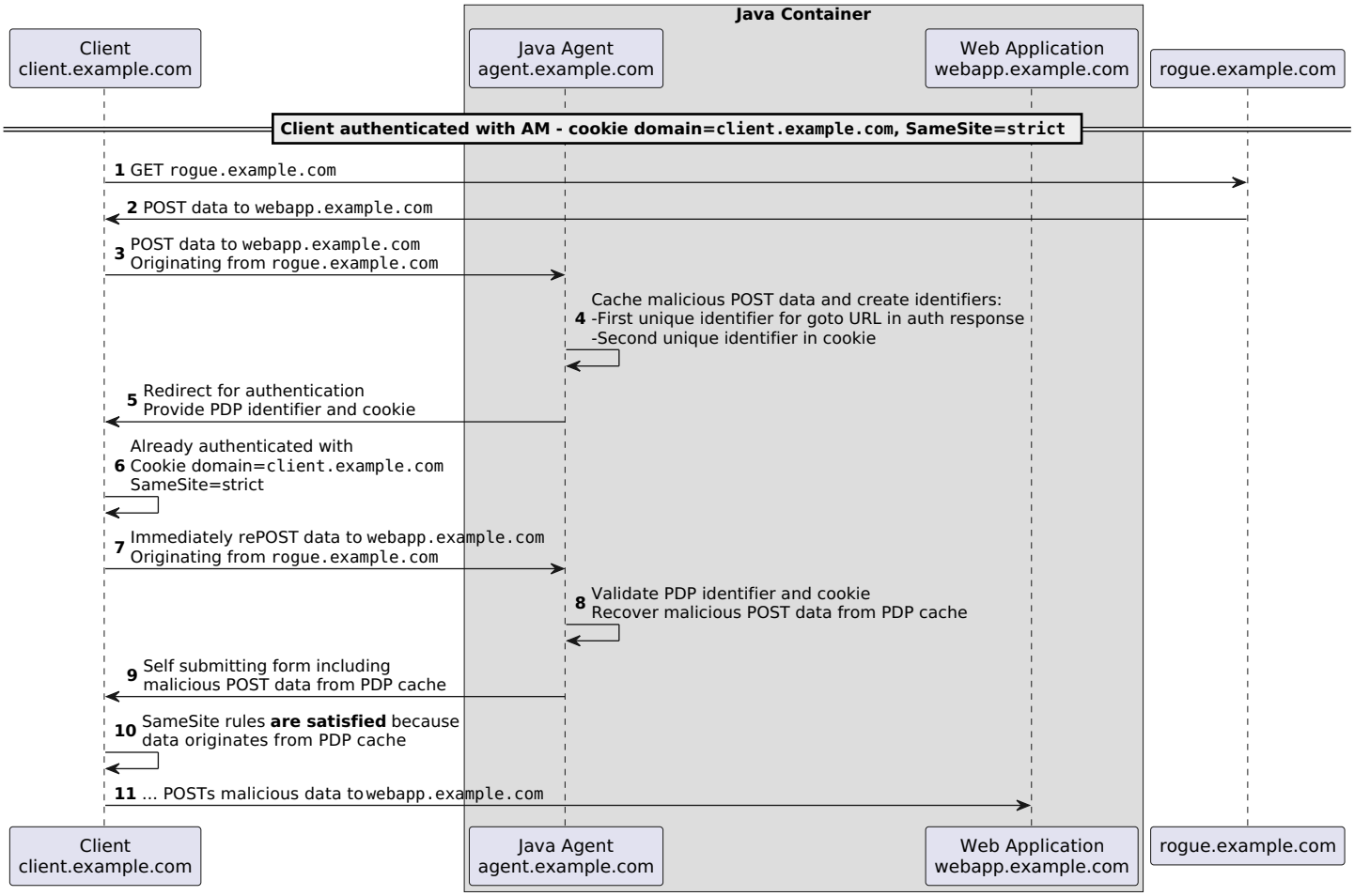
CSRF attack when POST data preservation is disabled

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, Java Agent redirects the user's browser to a login endpoint. The choice of endpoint and the parameters it receives is defined by the login redirect mode, *default* or *custom*.

Default login redirect

In default login redirect mode, the property Enable Custom Login Mode is always `false`. Depending on the configuration of login redirect properties, some endpoint parameters can be changed. For example, the agent can conditionally redirect a request to a specific realm or a different AM instance.

The `/oauth2/authorize` endpoint returns an OIDC ID token, and this is the only response the agent accepts.

Do **not** use default login redirect mode if session tokens for authentication and authorization are SSO tokens, even if you intend that the agent converts the SSO tokens into JWTs. Instead, use custom login redirect mode.

The following image shows the flow of data during a default login redirect:

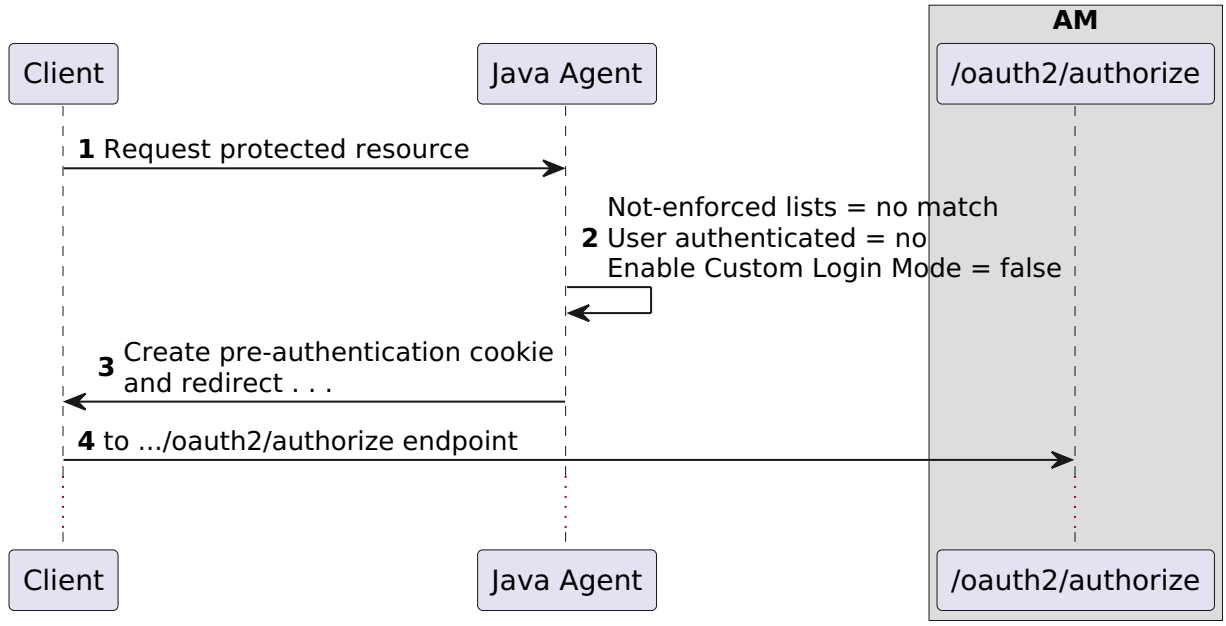


Figure 1. Data flow for default login redirect mode

Use the request domain to redirect login to a different realm

Set the following properties to redirect login to a different realm based on the domain of the request:

- Enable Custom Login Mode: Leave with the default value of `false`.
- AM Login URL List: Set to the URL of the login page, and specify the login realm as a parameter: `https://am.example.com:8443/am?realm=/myrealm`

The following image builds on figure 1, to configure AM Login URL List so that the agent redirects the user to log in to `myrealm` instead of the top-level realm.

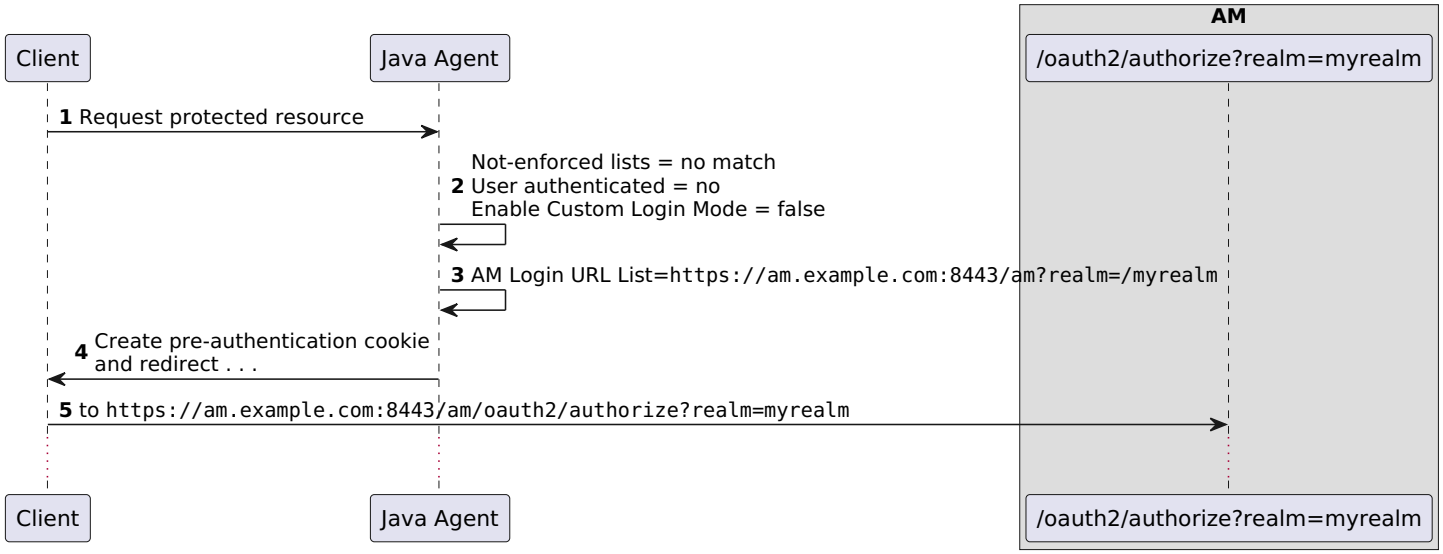


Figure 2. Data flow for default login redirect mode, where the user authenticates to a subrealm

Use the request domain to redirect login to a subrealm

Set the following properties to redirect a request to a login realm, based on the request domain:

- Enable Custom Login Mode: Leave with the default value of `false`.
- OAuth Login URL List: Map the request domain to the required login realm. When this property is set, the agent tries to match the request domain to the list of domains in this property. If there is a match, the agent redirects the user to log in at the matched URI.

The following image builds on figure 1, to configure OAuth Login URL List (`org.forgerock.agents.oauth.login.url.list`). Because the request is for a resource in `blue.example.com`, it is directed for authentication to the `blue` realm.

Figure 3. Data flow for default login redirect mode, where the user authenticates to a subrealm based on the request domain

Other requests are directed as follows:

- Requests for a resource in `red.example.com/ruby` are passed to the `oauth2/authorize` endpoint to log the user into the red realm.
- Requests for a resource in `red.example.com/yellow/` are passed to the `oauth2/authorize` endpoint to log the user into the yellow realm.
- Requests for a resource in an unmapped domain are passed to the `oauth2/authorize` endpoint to log the user in to the specified default realm.

Use the request domain to redirect login to different endpoints

In default login redirect mode, the agent can redirect requests to any AM instance supporting the `/oauth2/authorize` endpoint.

Set the following properties to redirect a request to a different OIDC endpoint, based on the request domain:

- Enable Custom Login Mode: Leave with the default value of `false`.
- OAuth Login URL List: Map the request domain to the required OIDC endpoint. When this property is set, the agent tries to match the request domain to the list of domains in this property. If there is a match, the agent redirects the user to log in at the matched OIDC endpoint.

The following image builds on figure 1, to configure OAuth Login URL List. Because the request is for a resource in `red.example.com/yellow`, it is directed for authentication to a different IDP.

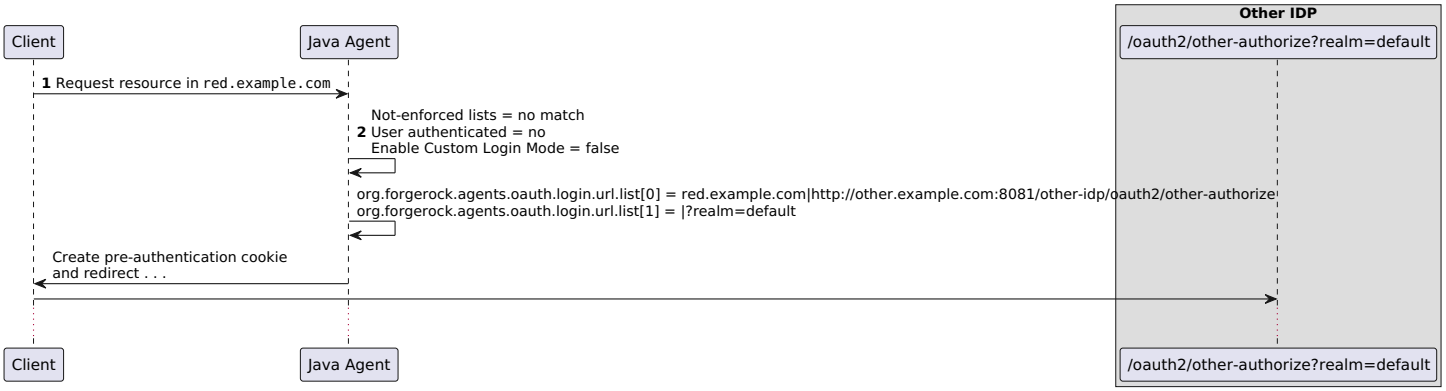


Figure 4. Data flow for default login redirect mode, where the user authenticates to an identity provider based on the request domain

Requests for a resource in an unmapped domain are passed to the AM `oauth2/authorize` endpoint, to log the user in to the specified default realm.

Custom login redirect

In custom login redirect mode, the agent is not confined to invoking a fixed endpoint in AM, but can redirect login anywhere. The agent handles JWTs or SSO tokens as session tokens for authentication and authorization.

Use custom login redirect mode for legacy deployments, where SSO tokens, instead of JWTs, are used for authentication and authorization. Otherwise, use default login redirect instead.

The property Enable Custom Login Mode is always `true`. Depending on the configuration of login redirect properties, the agent can:

- Convert SSO tokens into JWTs, through a direct "backdoor" call to AM
- Use caches to stop the SSO to JWT conversion from occurring more than once
- Leave SSO tokens unconverted

The following image shows the possible data flows for custom login redirect mode:

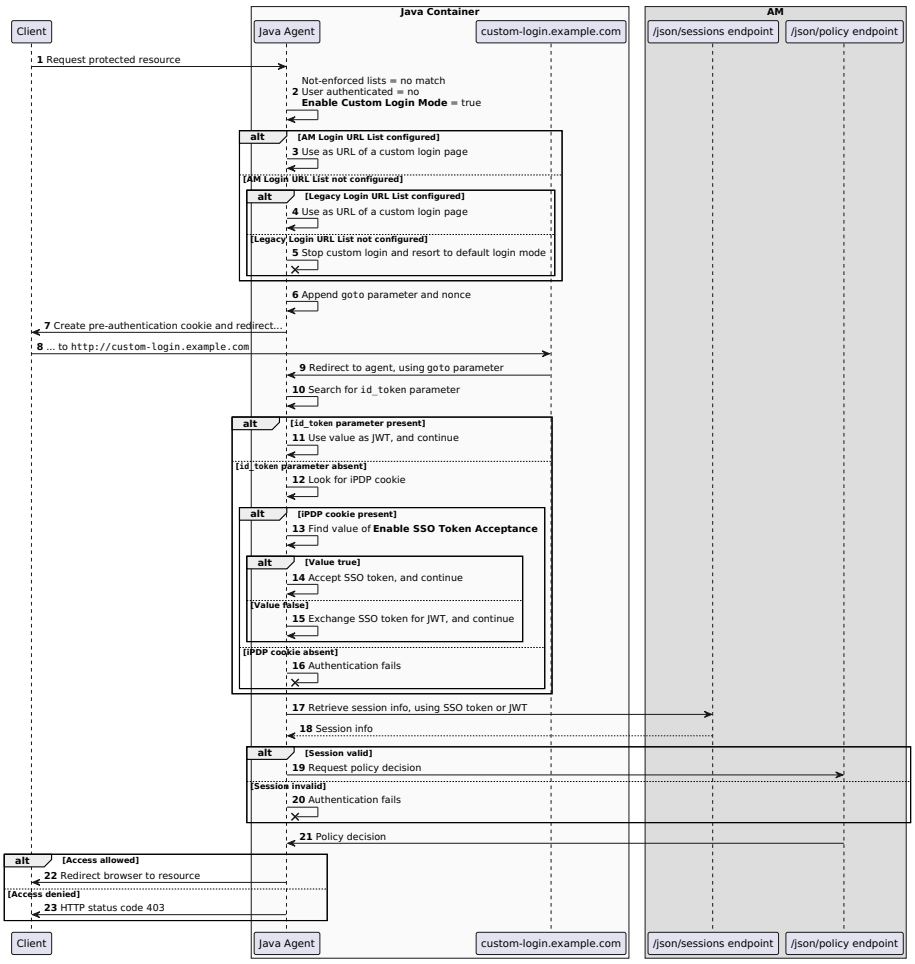


Figure 5. Data flow for customized login redirect

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 **Application** > **Agents** > **Java**, and select your Java Agent.
3. On the **AM Services** tab set the following properties:
    - Enable Custom Login Mode: Set to on
    - AM Login URL List: Set to the custom URL in step 1.

Redirect login to AM behind a firewall

When login must be completed in a network where AM is behind a firewall, set Public AM URL to a proxy which can access AM.

Limit the number of allowed redirect attempts

To mitigate the risk of infinite redirection loops, limit the number of redirects allowed for a browser session. After this number, the agent blocks the request.

Configure Redirect Attempt Limit, to specify a non-zero value. For example, if the limit is set to three, then the agent blocks the request on the fourth redirect.

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 redirection resource.

The resource to trigger logout can be the agent URL or a URL overridden by the configuration. The agent expects the logout to register session destruction with AM.

The agent maintains the user realm for each session, either by obtaining the realm info from the JWT, or by calling the `sessioninfo` endpoint (when SSO tokens are used). When the user logs out, the stored realm is passed to the logout endpoint automatically.

AM manages session cookies as follows, and the agent is responsible for destroying the cookies:

- From AM 7, AM places the session cookie in the `Authorization` header, prefixed with `X-Requester-Token`.
- Before AM 7, AM places the session cookie in the HTTP parameter `requester`.

If Convert SSO Tokens Into OIDC JWTs is `true`, the logout URL is invoked twice—once with the JWT, and again with the SSO token. If Enable SSO Token Acceptance is `true`, the logout URL can be invoked only by an SSO token.

Configure logout with the properties described in logout.

Trigger logout with a URL

Set the property Logout URI Map to specify a URL to trigger logout. When the URL is invoked, the agent kills the current session by invoking the AM REST logout endpoint or the endpoint configured by Conditional Logout URL List.

The URL is a dummy URL. Even if a resource exists at the URL, it is never accessed.

Log out of a specific web application

The following example triggers a logout from an application called `bank`, when the URL `http://app.example.com:80/mywebapp/bank/log-me-out` is invoked:

```
org.forgerock.agents.logout.endpoint.map[bank]=/bank/log-me-out
```

When a web application is specified, it must exist and the agent must have access to it. If the `bank` application in the above example doesn't exist, the web container throws an error.

Log out of all web applications

If a web application is not specified, the current sessions are killed for all web applications. The following examples trigger a logout from any application when the specified URL is invoked:

```
org.forgerock.agents.logout.endpoint.map=/agentapp/log-me-out
```

The agent must be able to access the context for the URL. For example, unless the agent is deployed in the root context, the following configuration fails:

```
org.forgerock.agents.logout.endpoint.map=/dummy-logout
```

Trigger logout with a parameter

Set the property Logout Request Parameter Map to specify a URL parameter to trigger logout. The agent searches every incoming request for the parameter. When the agent detects the parameter, it invokes AM to kill the current session for the specified web application.

To speed up the search for a logout parameter, set the property Enable Logout Introspection to `true`.

Log out of a specific web application

The following example triggers a logout from an application called `bank` when the request URL contains the parameter `log-out`:

```
org.forgerock.agents.logout.request.param.map[bank]=log-out
```

The request URL must contain the `log-out` parameter, but does not need to assign a value to the parameter. The following request URLs would trigger a logout for the previous configuration:

```
http://am.example.com:8080/protectedapp/index.html?log-out
http://am.example.com:8080/examples/index.html?examplelog-out=
```

Log out of all web applications

If a web application is not specified, the current sessions are killed for all web applications. The following example triggers a logout from any application when the request URL contains the parameter `logout`:

```
org.forgerock.agents.logout.request.param.map=logout
```

Conditionally log out to different URLs

Set Conditional Logout URL List to define URLs to which the agent can conditionally direct the user on logout.

Configure one or more conditions. The agent compares the request URL to each condition in the list to find the closest match. It evaluates conditions in order of length, starting with the longest, irrespective of their position in the list.

Depending on the value of the redirection URL, perform this additional configuration:

- If the URL doesn't perform a REST logout to AM, set Always invalidate sessions to `true`.

The agent additionally invokes the AM REST logout endpoint to invalidate the session.

- 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 PingOne Advanced Identity Cloud's Configure trusted URLs or AM's Configure trusted URLs.

In the following example, `example.com/path` is evaluated before `example.com`; the default condition is the shortest, and is evaluated last:

```
org.forgerock.agents.conditional.logout.url.list[0]=example.com|?additional=value
org.forgerock.agents.conditional.logout.url.list[1]=example.com/path|?one=red&two=green&three=blue
org.forgerock.agents.conditional.logout.url.list[2]=mybank.com|http://mybank.com/myapp/logout?param=override
org.forgerock.agents.conditional.logout.url.list[3]=|?alpha=beta
```

Using the above configuration, consider the following evaluations:

Request URL	Action
`http://example.com:9010/path/index.html`	The following parameter name:value pairs are added to the logout URL: `one:red`, `two:green`, and `three=blue`
`http://example.com:9010:/path/public/index.html`	
`http://example.com:9010:/index.html`	The following parameter name:value pair is added to the logout URL: `additional:value`
`https://mybank.com:443/path/index.html`	`http://mybank.com/myapp/logout` is used for logout, overriding the AM logout REST endpoint that the agent would use by default. The administrator is responsible for making sure that the overriding URL kills all tokens associated with login, but is not responsible for removing cookies containing either JWTs or SSO tokens.
Any URL that does not match on of the other conditions	Parameter `alpha:beta` added to logout URL

Redirect logout to a landing page

Set Logout Entry URI Map to redirect users to a specified resource after logout. Use this property for logout triggered by Logout URI Map or Logout Request Parameter Map.

The redirection resources can be HTML pages or JSP files. They are automatically added to the not-enforced list so that they can be accessed without authentication.

Depending on the type and value of a redirection resource, perform this additional configuration:

- If it is a URL that doesn't perform a REST logout to AM, set Always invalidate sessions to `true`.

  The agent additionally invokes the AM REST logout endpoint to invalidate the session.

- If it is a URL that 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 PingOne Advanced Identity Cloud's Configure trusted URLs or AM's Configure trusted URLs.

Configure a logout landing page for a specific web application

The following example directs requests to the `bank` application to `logout-page.html`, after logout:

```
org.forgerock.agents.logout.goto.map[bank]=/banking-app/logout-page.html
```

Configure a logout landing page for all web applications

To redirect requests for any web application, leave the web application name field empty, and set the logout URI as a specific URL. The following example directs all requests to `goodbye.html` after logout:

```
org.forgerock.agents.logout.goto.map=/agentapp/goodbye.html
```

Not-enforced rules

When an agent is configured to protect a web application, it protects **every** resource in that application. Each access to any resource incurs the overhead of a policy evaluation request to AM; access 100 resources, and you incur 100 requests to AM, with all of their associated network overhead.

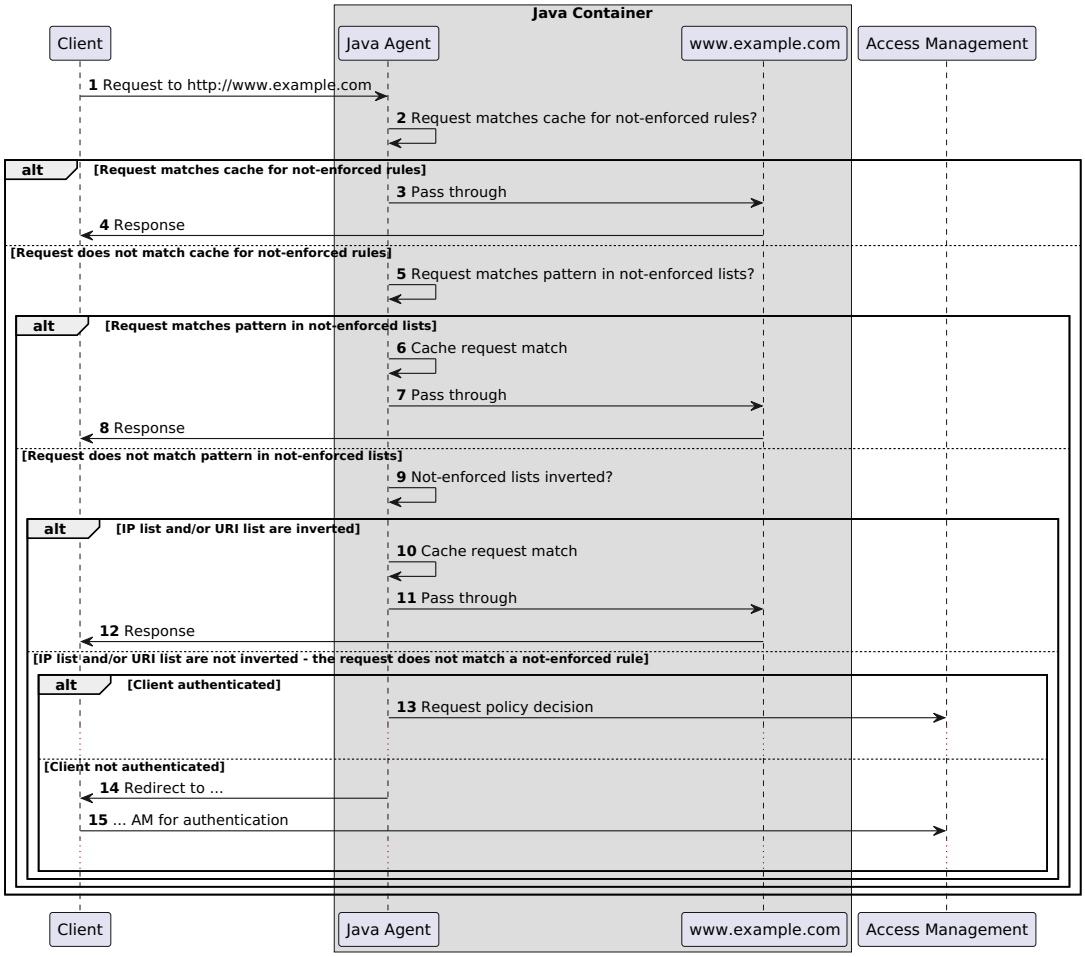
Some resources, such as the "public" directory of a web application, contain data that is not sensitive. It can be accessed by any, authenticated or unauthenticated, clients. The agent uses lists of *not-enforced rules* to identify these resources in the web application.

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 Java Agent evaluates not-enforced rules for a request, first searching for a match in the cache, then in the not-enforced lists:



1. A client requests a resource.

2-4. If the not-enforced URI or IP cache is enabled, the Java Agent checks whether the request matches any cached results. If the same request from the client previously matched a not-enforced rule, the Java Agent passes the request without requiring the client to authenticate.

5. If the caches are not enabled, or the request doesn't match a cached result, the Java Agent checks whether the request matches a rule in a not-enforced list.

The Java Agent evaluates every rule in the lists in order, until it finds the first match. When it finds a match, it stops evaluating, and does not consider other rules further down the list even if they provide a better match. Take care to place your most specific rules at or near the beginning of the list.

6-8. The Java Agent caches the result and passes the request without requiring the client to authenticate.

9-14. If the request doesn't match a rule in a not-enforced list, the Java Agent checks whether rules are inverted, and responds as follows:

Not-enforced URI rules	Not-enforced IP rules	Pass request without requiring authentication?
Inverted	Inverted	Yes

Not-enforced URI rules	Not-enforced IP rules	Pass request without requiring authentication?
Not inverted	Not inverted	No
Inverted	Not inverted	No
Not inverted	Inverted	No

Configure not-enforced rules

Configure not-enforced rules by using the properties listed in Not-enforced rules in the *Properties reference*, or on the [path]**Application** tab of the AM admin UI. Configure the following lists of not-enforced rules:

Not-enforced URI rules

Rules in Not-Enforced URIs allow access to resources, such as images, stylesheets, or the HTML pages that provide the public front end of your site.

Not-enforced IP rules

Rules in Not-Enforced Client IP List allow access to your site from an administrative IP address, an internal network range, or a search engine.

Compound not-enforced URI and IP rules

Allow access based on a combination of resources and IPs.

When there are multiple lists of rules, the agent evaluates them in the following order:

Order	Rule type	Rule includes requirements for Cookie values or Header values	Rule uses the `DENY` keyword to Deny access
1	Compound	Yes	Yes
2	Compound	Yes	No
3	Compound	No	Yes
4	Compound	No	No
5	IP	Yes	Yes
6	IP	Yes	No
7	IP	No	Yes
8	IP	No	No
9	URL	Yes	Yes

Order	Rule type	Rule includes requirements for Cookie values or Header values	Rule uses the DENY keyword to Deny access
10	URL	Yes	No
11	URL	No	Yes
12	URL	No	No

Conventions for not-enforced rules

Use the following conventions to define not-enforced URI rules and not-enforced IP rules.

Path normalization

Java Agent normalizes resource paths *before* applying not-enforced rules.

This affects the following sequences, for example:

- `%2F` becomes `/`
- `%5C` becomes `/`

Invert rules

Invert specific rules

> **Important**
>
> The `NOT` keyword is overridden by the `DENY` keyword. Refer to Deny access.

Invert any rule in a not-enforced list by preceding it with the keyword `NOT`, separated by a space (blank) character.

In the following example, requests for a `.jpg` file in the `/private` URI require authentication:

```
NOT /private/*.jpg
```

In the following example, the agent enforces authentication and requests policy evaluations for any request from the network specified by the `192.168.1.0/24` CIDR notation:

```
NOT 192.168.1.0/24
```

Invert all rules

> **Important**
>
> For security considerations, do not invert all rules. Instead, ForgeRock recommends using the NOT keyword to invert specific rules.

> **Important**
>
> The `NOT` keyword and the properties in this section are overridden by the `DENY` keyword. Refer to Deny access.

Invert all rules by setting Invert Not-Enforced IPs or Invert Not-Enforced URIs to `true`.

Wildcards

For more information about using wildcards, refer to Wildcards.

Note that trailing forward slashes are not recognized as part of a resource name. Therefore, `/images//`, `/images/`, and `/images` are equivalent.

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 character (`\`) 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 a regular expression.

The following table gives examples of the multi-level wildcard `*` in rules defined in Not-Enforced Client IP List:

Multi-level wildcard for not-enforced IP rules

Rule	Matches request IP	Does not match request IP
`192.168.1.*`	`192.168.1.0` `192.168.1.0/24`	`192.168.0.1`

The following table gives examples of the multi-level wildcard `*` in rules defined in Not-Enforced URIs:

Multi-level wildcard for not-enforced URI rules

Rule	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`

Rule	Matches request URL	Does not match request URL

`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`

`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 multiple levels in a URL.
- Cannot be escaped. Therefore, the backslash character (\) 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 a regular expression.

The following table gives examples of the one-level wildcard `-*-` in rules defined in Not-Enforced URIs:

One-level wildcard for not-enforced URI rules

Rule	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/cd/`
		`http://A-examp.com:8080/b`
		`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 URIs, 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:

```
/customers/*?*member_level=*&location=*
```

The following 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
```

Deny access

Use the `DENY` keyword with a regular expression or classic pattern to prevent access either to a resource or from an IP address. If the request matches the resource path or IP address, Java Agent denies access with an HTTP 403 Forbidden status.

When the `DENY` keyword is used, access to a resource and/or access from an IP address is *always* denied. If the `NOT` keyword or the following properties are used with the `DENY` keyword, they are ignored: Invert Not-Enforced IPs or Invert Not-Enforced URIs.

Consider the following examples:

- Deny access to requests from the specified IP address:

  ```
  DENY 155.251.79.32
  ```

- Deny access to all .jpg files:

  ```
  DENY /*.jpg
  ```

  Note here that the use of `DENY` causes `NOT` to be ignored:

  ```
  NOT,DENY /*.jpg
  ```

- `DENY` access to incoming URLs containing dubious characters, such as percent characters remaining in the normalized path. Java Agent converts `%2F` to `/` and `%5C` to `\` before applying not-enforced rules:

  ```
  DENY /*%*
  ```

Regular expressions

> **Important**
>
> The Java Agent uses regular expression matching from the JDK. Make sure your expressions are evaluated in a way that is consistent with this.

Add the keyword `REGEX` or `REGEXP` followed by a blank (space) character before the URI or IP address. For example:

```
REGEX https?://www\.example\.com/([^/])+/.*\.jpg
```

```
REGEX 192\.168\.10\.\d+
```

Consider the following points when using regular expressions:

- Wildcards and regular expressions cannot be used in the same rule.
- Using netmask CIDR notation or IP address ranges and regular expressions is not supported. However, you can create a regular expression that matches a range of IP addresses, such as:

  ```
  REGEX 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

Add one or more of the following keywords to the not-enforced rule to apply it when the incoming request uses a specific HTTP method: `GET`, `HEAD`, `POST`, `PUT`, `PATCH`, `DELETE`, `OPTIONS`, `TRACE`.

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 require authentication for any request method to `192.168.10.*`:

```
192.168.10.*
```

In the following example, the agent does not enforce authentication or request policy evaluations for GET requests to `/public`, but does for other HTTP methods:

```
GET /public/*
```

To specify a list of methods, add a comma-delimited list of methods, followed by a blank (space) character before the item to match.

```
GET,POST /public/*
GET,POST,PUT /examples/notenforced/*.jpg
GET,REGEX https?://www\.example\.com/([^/])+/.*\.jpg
```

```
NOT,GET,REGEX 192\.168\.10\.\d+
POST 192.168.10.*
GET 192.168.10.1-192.168.10.254 192.168.0.1
POST,PUT 192.168.1.0/24
```

To invert a method, precede it with an exclamation point `!` character. In the following examples, the agent enforces authentication and requests policy evaluations for `POST` requests, but not for other HTTPS methods:

```
!POST /public/*
```

```
!POST 192.168.1.0/24
```

Unrecognized keywords in a rule are ignored and do not invalidate the rest of the rule.

Cookie values

Use the following syntax to require the incoming request to have a named cookie with a specified value:

```
COOKIE(Name/Value/Modifiers) Not Enforced URIs
```

```
COOKIE(Name/Value/Modifiers) Not Enforced IPs
```

- Name: Cookie name
- Value: Cookie value
- Modifiers: One or more modifiers to change the lookup method:
  - `c`: (For not-enforced URI rules only) Perform a case-insensitive search for the cookie name. By default, the search is case-sensitive.
  - `i`: Perform a case-insensitive search for the cookie value. By default, the search is case-sensitive.
  - `r`: Treat the string in Value as a regular expression.

The following example does not require authentication for requests to `/private/admin/images/`, when the request contains a cookie with the case-insensitive name `login_result`, and case-insensitive value `valid`:

```
COOKIE(login_result/valid/ci) /private/admin/images/*
```

Because the search is case-insensitive, the following example is equivalent:

```
COOKIE(Login_result/VALID/ci) /private/admin/images/*
```

In the following example, the agent does not enforce authentication or request policy evaluations for requests to `192.168.*`, when the request contains a cookie with the case-sensitive name `login_result` and the case-insensitive value `VALID`:

```
COOKIE(login_result/VALID/i) 192.168.*
```

Combine cookie filters with other filters, such as HTTP methods. Combining a `HEADER` and `COOKIE` expression in the same rule implies a logical AND; both expressions must match in order to apply. To apply the rules as a logical OR, create two separate rules.

In the following example, the agent does not enforce authentication or request policy evaluations for GET, POST, and PUT requests to the `/other/records/` folder, when the request contains a cookie with the case-sensitive name `internal`, and a case-insensitive value that ends with `.*ID`:

```
GET,POST,COOKIE(internal/.*ID/ri),PUT /other/records/*.html
```

In the following example, the agent does not enforce authentication or request policy evaluations for `GET`, `POST`, and `PUT` HTTP requests from the client IP range `192.168.*`, when the request contains a cookie with the case-sensitive name `internal`, and a case-sensitive value that ends with `.*ID`:

```
GET,POST,COOKIE(internal/.*ID/r),PUT 192.168.*
```

Header values

Use the following syntax to require the incoming request to have a named header with a specified value:

```
HEADER(Name/Value/Modifiers) Not Enforced URIs
```

```
HEADER(Name/Value/Modifiers) Not Enforced IPs
```

- Name: Header name.
- Value: Header value to search for.
- Modifiers
  - `i`: Perform a case-insensitive search for the header value. By default, the search is case-sensitive.
  - `r`: Treat the string in Value as a regular expression.

In the following example, the agent does not enforce authentication or request policy evaluations for access to .txt files in `/yearly/2021/` when the request contains a header with the case-sensitive name `ID`, and a case-insensitive value `validated`

```
HEADER(ID/validated/i) /yearly/2021/*.txt
```

In the following example, the agent does not enforce authentication or request policy evaluations for access to the IP range `192.168.*` when the request contains a header with the case-sensitive name `ID`, and a case-insensitive value `validated`:

```
HEADER(ID/validated/i) 192.168.*
```

Combine cookie filters with other filters, such as HTTP methods. Combining a `HEADER` and `COOKIE` expression in the same rule implies a logical AND; both expressions must match. To apply the rules as a logical OR, create two separate rules.

In the following example, the agent does not enforce authentication or request policy evaluations for `GET`, `POST`, and `PUT` requests to HTML resources in the `/other/records/` folder when the request contains a header with the case-insensitive name `internal`, and a case-insensitive value that ends with `.*ID`:

```
GET,POST,HEADER(internal/.*ID/ri),PUT /other/records/*.html
```

Compound rules

Configure compound not-enforced rules to combine not-enforced URI and IP rules in a single rule.

Configure rules in either Not-Enforced Client IP List or Not-Enforced URIs, using an IP rule or list of IP rules, a delimiter, and an URI rule or list of URI rules.

In the following example, the agent does not enforce authentication or request policy evaluations for HTTP requests from the IP range `192.168.1.1-192.168.4.3` to any file in the `/images` URI:

```
192.168.1.1-192.168.4.3 | /images/*
```

Consider the following points for compound rules:

- Place keywords, such as HTTP methods, `NOT`, and `REGEX`, at the beginning of the compound rule. Keywords affect both the IP and the URI rules.

  In the following example, the agent does not enforce authentication or request policy evaluations for `GET` or `POST` HTTP requests from the IP range `192.168.1.1-192.168.4.3`, to any file (`*`) in the `/images` URI.

  ```
  GET,POST 192.168.1.1-192.168.4.3 | /images/*
  ```

  In the following example, the agent enforces authentication and requests policy evaluations for any request to a method except `POST`, from any IP address in the `192.168.1` subnet, to any file in the `/private` URI.

  ```
  NOT,!POST 192.168.1.* | /private/*
  ```

- Check that both sides of a rule using the `REGEX` keyword can be parsed as a regular expression.

  In the following example, the delimiter is `&&`, because the `|` character can lead to invalid regular expressions:

  ```
  POST,REGEX 192\.168\.10\.(10|\d) && \/images\/([^/])+\.*\.jpg
  ```

  For information about configuring a different delimiter, refer to Not-Enforced Compound Rule Separator.

- The agent caches hits and misses for each resource accessed.

  Caching is enabled if either Enable Not-Enforced IP Cache or Enable Not-Enforced URIs Cache is `true`.

  The cache size takes the biggest value of Max Entries in Not-Enforced IP Cache or Max Entries in Not-Enforced URI Cache.

Extended characters

URLs defined in Not-Enforced URIs can contain any number of extended ASCII characters. The agent container automatically percent-encodes extended characters, before the agent is called.

Extended characters in the resource path of a not-enforced rule

By default, Java Agent uses UTF-8 to percent-encode extended characters in the resource paths of not-enforced rules. To change the character encoding, set Container Character Encoding.

In the following example, the agent does not enforce authentication or request policy evaluation for HTTP requests to the URL `http://www.example.com/forstå`:

```
org.forgerock.agents.notenforced.uri.list=http://www.example.com/forstå/*
```

Note how the extended ASCII character `å` can be entered without encoding.

Extended characters in HTTP query parameters of a not-enforced rule

By default, Java Agent uses ISO-8859-1 to encode extended characters in HTTP query parameters of not-enforced rules. To change the character encoding, set Container Parameter Encoding.

Continuous security

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

When the Java Agent requests a policy decision from AM, it can include this information in an *environment map*, a set of name/value pairs that describe the request IP and DNS name, along with other, optional, information.

In Java Agent, use *continuous security* to configure an environment map. 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 agent configuration properties, refer to Continuous security. For information about server-side authorization scripts, refer to Scripting a policy condition in AM's *Authorization guide*.

Environment maps with customizable keys

In Java Agent, use the continuous security properties Client Hostname Header and Client IP Address Header to configure an environment map with custom keys.

The environment map has the following parts:

requestIp

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

- If Client IP Address Header is configured, the Java Agent extracts the IP address from the header.
- Otherwise, it uses the Java function `HttpServletRequest.getRemoteAddr` to determine the 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, the Java Agent extracts the host name from the header.
- Otherwise, it uses the Java function `HttpServletRequest.getRemoteHost` to determine the host name address.

This entry is always created in the map.

Other variable names

An array of cookie or header values, configured by the continuous security properties Client Hostname Header and Client IP Address Header.

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.agents.continuous.security.cookies.map[ssid]=mySsid
org.forgerock.agents.continuous.security.headers.map[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
```

Environment maps with fixed keys

In Java Agent, use the following properties to configure an environment map with fixed keys:

- GET Parameter List for URL Policy Env
- POST Parameter List for URL Policy Env
- JSession Parameter List for URL Policy Env

Caching

Java Agent allocates memory from the Java heap space in the web container to the caches described in this section.

Configuration cache

When the agent starts up in remote configuration mode, it retrieves a copy of the agent profile from AM, and stores it in the cache. The cached information is valid until one of the following events occurs:

- AM notifies the agent of changes to hot-swappable agent configuration properties. The agent flushes the configuration cache and rereads the agent profile from AM.
- The agent restarts.
- The agent rereads the configuration from AM or from local files at the frequency specified by Configuration Reload Interval.

If the reload interval is disabled, and notifications are disabled, the cached configuration remains valid until the agent restarts.

Session cache

After authentication, AM presents the client with a JWT, containing session information. The agent stores part of that session information in the cache.

A session stored in the session cache is valid until one of the following events occur:

- The session contained in the JWT expires.
- The client logs out from AM, and session notifications are enabled.
- The session reaches the expiration time specified by Session Cache TTL.

Policy decision cache

When a client attempts to access a protected resource, the agent checks whether there is a policy decision cached for the resource:

- If the client session is valid, the agent requests a policy decision from AM and then enforces it.
- If the client 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, the agent requests policy decision from AM and enforces it.

Policy decisions are valid in the cache until one of the following events occur:

Session and policy validity in cache

Event	What is invalidated?
Session contained in the JWT 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?
Policy decision reaches the expiration time specified by Policy Cache TTL	Policy decision
Administrator makes a change to policy configuration (and policy notifications are enabled)	All sessions and all policy decisions

Important

A Java Agent that loses connectivity to 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.

Not-enforced lists hit and miss caches

The first time the agent receives a request for a resource, it matches the request and the client's IP address against the rules specified in the not-enforced lists.

Java Agent maintains a cache of hit-and-miss for each of the not-enforced lists specified in Not-enforced rules.

To speed up future requests, the agent stores whether the resource hit or missed not-enforced rules in the corresponding caches. Therefore, if a request for the same resource reaches the agent again, the agent replays the result of the rule evaluations stored in the caches, instead of re-evaluating the request.

Entries stored in the hit and miss caches do not expire unless AM notifies the agent about configuration changes in the not-enforced lists.

POST data preservation cache

When POST data preservation is enabled, the agent caches HTML form data submitted as an HTTP POST by unauthenticated clients.

The POST data expires either when the client recovers the information from the cache or after the time interval specified in POST Data Preservation Cache TTL.

For more information, refer to POST data preservation.

OpenID Connect JWT cache

Decoding JWTs into JSON objects is a CPU-intensive operation. To reduce the amount of processing required on each request, agents cache decoded JWTs.

When an agent receives a request for a resource, it passes the JWT through a fast hashing algorithm that creates a 128-bit hash unique for that JWT. Then the agent determines if the hash is in the JWT cache. One of the following scenarios occur:

- The hash is in the cache. The agent retrieves the decoded JWT from the cache and continues processing the request.
- The hash is not in the cache. The agent decodes the JWT and stores it and its hash in the cache. Then it continues processing the request.

  JWTs in the cache expire after the time interval specified by JWT Cache TTL.

Attribute fetch modes

For information about properties to configure attribute fetching, refer to Attributes.

Java Agent can fetch and inject user information into HTTP headers, request objects, and cookies, and pass them on to client web applications. The client web applications can personalize content using these attributes in their web pages or responses.

You can configure the type of attributes to fetch, and map the attribute names used on AM to the values used in the containers. The agent securely fetches the user and session data from the authenticated user, as well as policy response attributes.

Autonomous mode

In autonomous mode, the agent operates independently of AM, without needing to contact an AM instance. Agents allow access to resources as defined in not-enforced lists; otherwise, they deny access.

Agents evaluate not-enforced rules using the following features:

- URLs, IP addresses, IP address ranges, and compound rules.
- Rules applied to specific HTTP methods.
- Inverted not-enforced rules, by using properties.
- Inverted not-enforced rules, by using inline logical operators.
- Rules that use regular expressions.
- Rules applied in the presence of named cookies with specified values.

Because the agent does not attempt to contact AM, the following functionality is not available in autonomous mode:

- Notifications
- Remote auditing
- Profile attributes
- Session attributes
- Response attributes
- Continuous security

To enable autonomous mode, in the bootstrap properties file, `AgentBootstrap.properties`, set Autonomous mode to `true`, and restart the Java container where the agent is installed.

> **Important**
>
> Because the agent does not contact AM when it starts in autonomous mode, the value of Location of Agent Configuration Repository must be `LOCAL`.

FQDN checking

When FQDN checking is enabled, the agent can redirect requests to different domains, depending on the hostname of the request. Use this feature in environments where the request hostname can be virtual, invalid, or partial.

FQDN checking requires Enable FQDN Checking to be `true`, Default FQDN to be set to a suitable value, and optionally, FQDN Map to be set to suitable default FQDN.

When FQDN Map is configured, the agent maintains the following maps:

- Map 1:
  - Key: Incoming hostname without wildcards.
  - Value: Outgoing hostname.
- Map 2:
  - Key: Incoming hostname with wildcards `*` and `?`.
  - Value: Outgoing hostname.

Map keys are case insensitive. Incoming hostnames are converted to lowercase before the agent maps them, and the agent automatically converts uppercase keys and values to lowercase before mapping.

The agent maps FQDNs as follows:

1. Searches map 1 for the incoming hostname. If there is a match, the agent redirects the request to the mapped value.
2. Searches map 2 for a pattern that matches the incoming hostname, iterating through the entries in random order. If there is a match, the agent redirects the request to the mapped value.
3. Redirects the request to the value in Default FQDN.

Examples

The following example configuration and requests illustrate how the agent checks and remaps FQDNs:

Configuration

- Enable FQDN Checking: `org.forgerock.agents.fqdn.check.enabled=true`
- Default FQDN: `org.forgerock.agents.fqdn.default=agent.defaulttest.me`
- FQDN Map:
  - Map 1

    ```
    org.forgerock.agents.fqdn.map[agent]=agent.localtest.me
    org.forgerock.agents.fqdn.map[agent.virtualtest.me]=virtual-host.localtest.me
    ```

  - Map 2

    ```
    org.forgerock.agents.fqdn.map[agent-*.localtest.me]=agent.localtest.me
    ```

Example requests

- `https://agent.localtest.me/app` : Does not match any mapping, so the agent redirects it to the default FQDN `https://agent.defaulttest.me/app` .
- `https://agent/app` : The request URL matches the first mapping in map 1, so the agent redirects it to `https://agent.localtest.me/app` .
- `https://AGENT/app` : The request URL matches the first mapping in map 1, because incoming hostnames are converted to lower-case before the agent maps them. The agent redirects the request to `https://agent.localtest.me/app` .
- `https://agent.virtualtest.me/app` : The request URL matches the second mapping in map 1, so the agent redirects it to the virtual host `https://virtual-host.localtest.me/app` .
- `https://agent-123.localtest.me/app` : The request URL matches the mapping in map 2, so the agent redirects it to `https://agent.localtest.me/app` .

Cookie reset

Java Agent can reset cookies in its own domain before redirecting the client for login, and when the client logs out.

Pre-authentication cookies are reset automatically after successful authentication. Authentication cookies are reset automatically on logout. This section describes how to manage reset of other cookies.

To enable cookie reset, set Cookie Reset to `true` . The agent resets the cookies in the response before redirecting the client for login, and when the client logs out.

To reset specific cookies, add them to the list in Reset Cookie List. The agent searches for the cookie name using a case-sensitive search. If it finds a match, the cookie is reset. Otherwise, the agent searches again using a case-insensitive search. If it then finds a match, the cookie is reset and a warning is issued to the logs.

When profile or session attributes are stored in cookies (either Profile Attribute Fetch Mode or Session Attribute Fetch Mode has the value `HTTP_COOKIE`), cookie reset is enabled automatically and cannot be disabled. The agent resets the profile and session attributes cookies and the cookies in the Reset Cookie List.

When response attribute are stored in cookies (Response Attribute Fetch Mode has the value `HTTP_COOKIE`), the agent does not reset them automatically. To prevent a build up of response attribute cookies, consider adding them to the Reset Cookie List.

```
org.forgerock.agents.cookie.reset.name.list[0]=response-attribute-cookie-name1
org.forgerock.agents.cookie.reset.name.list[1]=response-attribute-cookie-name2
```

To specify the paths for which cookies named in Reset Cookie List are used after reset, set the Reset Cookie Path Map.

Consider enabling cookie reset when the agent is deployed with parallel authentication mechanisms. Resetting cookies from one authentication mechanism before redirecting clients to log in with another mechanism helps prevent issues on the new login site.

Authentication failure

When a client does not present a valid SSO token with a request, Java Agent redirects the client to login. If the client then fails to authenticate, by default, the agent takes the following steps:

1. Redirects the request to the URL defined by Authentication Fail URL.
2. If that property is not set, redirects the request to the URL defined by Goto URL.
3. If neither property is set, returns an HTTP 400.

To limit the amount of information available to malicious users, by default, the agent returns an HTTP 400 for all authentication failures, regardless of the reason.

If, for example the agent returns an "unknown user" message, malicious users can use that information to try with different usernames until the error message changes to, for example, "wrong password".

The following table summarizes possible reasons for the agent to return an HTTP 400:

Reason code	Meaning
`AUTHN_BOOKKEEPING_COOKIE_MISSING`	The agent cannot find the authentication tracking cookie, defined in Pre-Authentication Cookie Name. This error can happen if the user successfully authenticates, but clicks the back button of the browser to return to the previous page.
`NONCE_MISSING`	The agent found the authentication tracking cookie, but it cannot find the unique identifier of the authentication request inside the cookie. This error can happen if the user successfully authenticates, but clicks the back button of the browser to return to the previous page.
`BAD_AUDIENCE`	The audience in the JWT does not correspond to the audience in the cookie entry. This error can happen if all agents working in a cluster do not have the same Agent Profile Name.
`NO_TOKEN`	The agent cannot find the session ID token.
`TOKEN_EXPIRED`	The agent found the session ID token, but it is past its expiry date.
`AM_SAYS_INVALID`	The agent found the session ID token, the expiry time is correct, but AM returns that the ID token is invalid.
`JWT_INVALID`	The agent found the session ID token, but cannot parse it.
`EXCEPTION`	The agent found the session ID token, but threw an exception while parsing it. Alternatively, the agent cannot connect to AM to validate the ID token, maybe due to a network outage.

Manage notifications for authentication failure

An HTTP 400 message is not always helpful for debugging the agent flow or when another web application depends on the error message. To change the way the agent responds to authentication failure, configure the following properties:

- Authentication Fail URL, to redirect the uses to a specific URL or URI. Use this property to control the message the agent displays to the client.
- Authentication Fail Reason Parameter Name, to send the reason for authentication failure in a named query parameter.
- Authentication Fail Reason Parameter Value Map, to map the reason for authentication failure. Use this property to hide the reason for authentication failure from malicious users, or to map it to something that is meaningful inside your organization.

Limit the number of failed login attempts

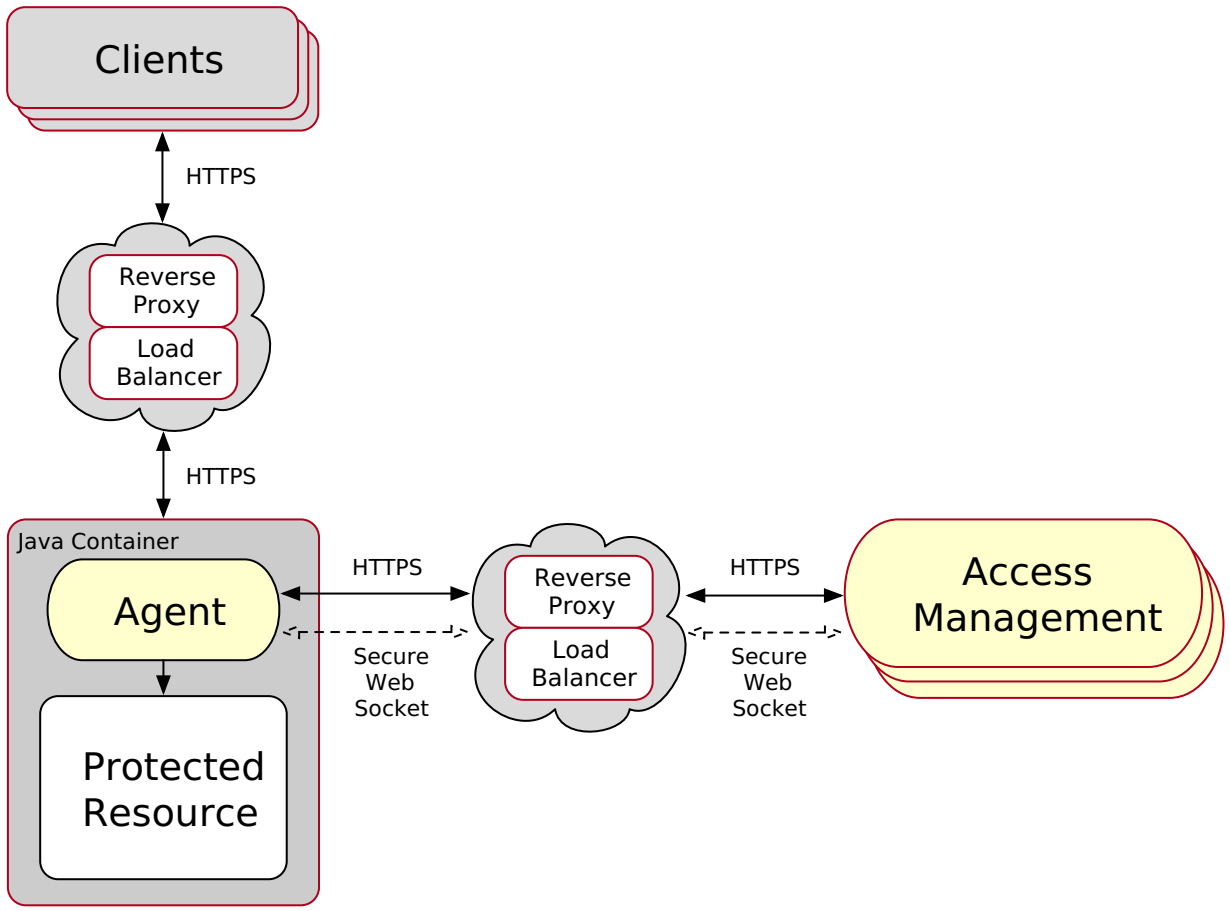
> **Important**
>
> This feature is deprecated. For more information, refer to the Deprecated section of the *Release Notes*.

To mitigate the risk of brute force attacks, limit the number of failed login attempts that are allowed during a browser session. After this number, the agent blocks requests from the user.

Configure Login Attempt Limit (deprecated), to specify a non-zero value. For example, if the limit is three, then the agent blocks the fourth and subsequent login requests.

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:



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 - load balancer/reverse proxy - AM

When a reverse proxy is situated between the agent and AM, it protects the AM APIs.

When a load balancer is situated between the agent and AM, it regulates the load between different instances of AM.

Consider the points in this section when installing Java Agent in an environment where AM is behind a load balancer or a reverse proxy.

Pre-authentication cookie signing

During installation, the agent requests the path to a file containing the cookie signing key, and then uses the key to configure the property `org.forgerock.agents.cookie.signing.value` in AgentKey.properties. If the path is empty, cookie signing is disabled.

The key must be at least 64 characters long. If it is shorter, the agent rejects it and leaves cookies unsigned. For security, use a key of at least 80 characters.

In deployments with multiple agent instances, use the same cookie signing key for each instance. Share a cookie signing key between agent instances as follows:

- If agent instances can share the signing key file, specify the same file for each agent installation.
- If agent instances are in remote configuration mode, set the property `org.forgerock.agents.cookie.signing.value` in the **Advanced** tab of the AM admin UI.

  When storing shared keys in AM Secure communication between the agent and AM.
- Manually update the agent configuration, as defined in the following procedure:

1. Install the first agent instance.
2. Note the value of `org.forgerock.agents.cookie.signing.value` in the AgentKey.properties file.
3. Install the next agent instance, and then replace the value of `org.forgerock.agents.cookie.signing.value` in AgentKey.properties with the value from the first instance.
4. Restart the instance.

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.

Do the following to prevent problems during installation, or with redirection using the `goto` parameter:

- 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, refer to 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 the agents are configured with multiple agent profiles, configure sticky load balancing. This is because the agent profile name is contained in the OpenID Connect JWT, used by the agent and AM for communication. Without session stickiness, there is no way to make sure that the appropriate JWT ends in the appropriate agent instance.
- 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 the 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 an example of how to configure Apache HTTP as a reverse proxy, refer to Configure an Apache HTTP Server as a reverse proxy.

Configure 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.

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** > **Create**, leaving the fields empty.
4. Configure the service with the following properties:
    - **Base URL Source**: X-Forwarded-* headers

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

      Leave the other fields empty.

      For more information, refer to Base URL source in AM's *Reference*.
5. Save your changes.

Agent - load balancer/reverse proxy - client

When a reverse proxy is situated between the agent and client, it renders anonymous the client traffic that enters the network.

When a load balancer is situated between the agent and client, it regulates the load between the agents and the containers.

Consider the points in this section when installing Java Agent in an environment where clients are behind a load balancer or a reverse proxy:

Forward 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.

Use sticky load balancing with POST data preservation

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

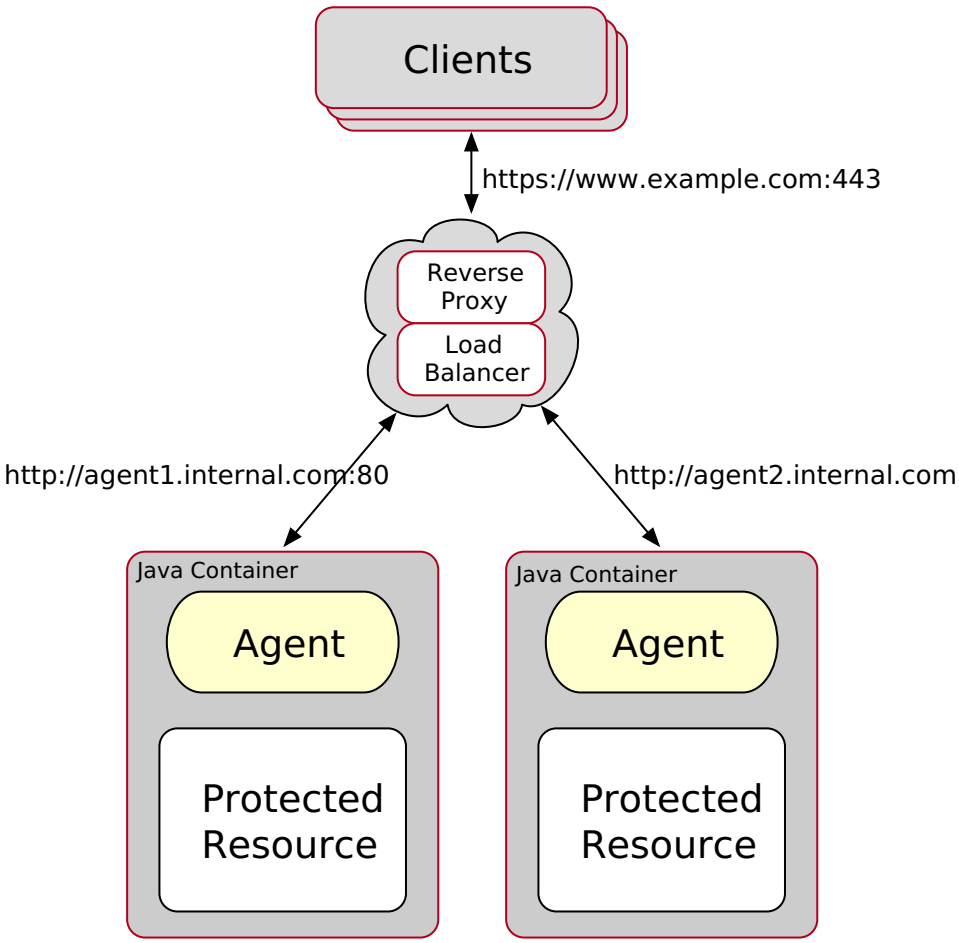
Agents provide properties to set either sticky cookie or URL query string for load balancers and reverse proxies.

For more information, refer to Configuring POST Data Preservation for Load Balancers or Reverse Proxies.

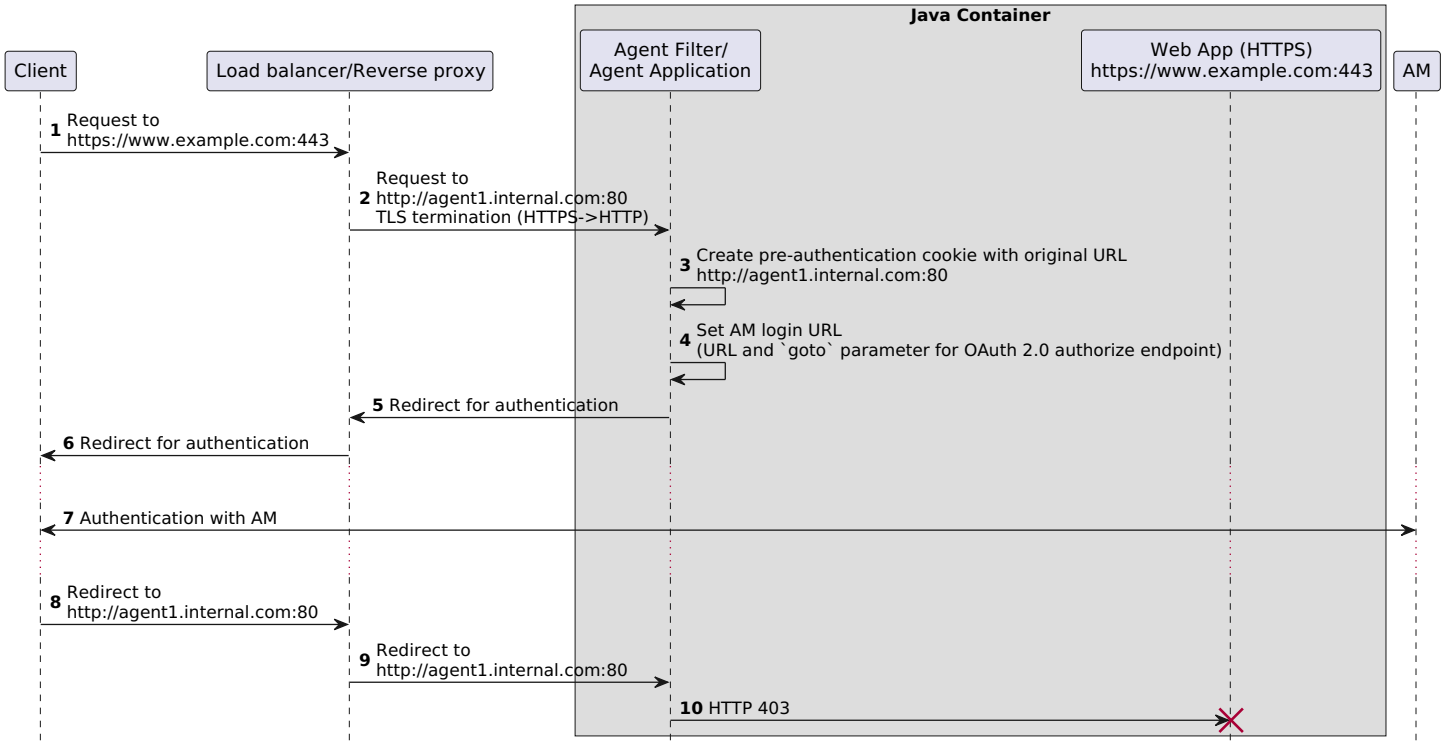
Override protocol, host, and port after TLS offloading

The load balancer or reverse proxy performs TLS offloading, terminating the TLS traffic and converting the requests reaching the Java container to HTTP. This reduces the load on the protected containers, because the public key is usually processed by a hardware accelerator.

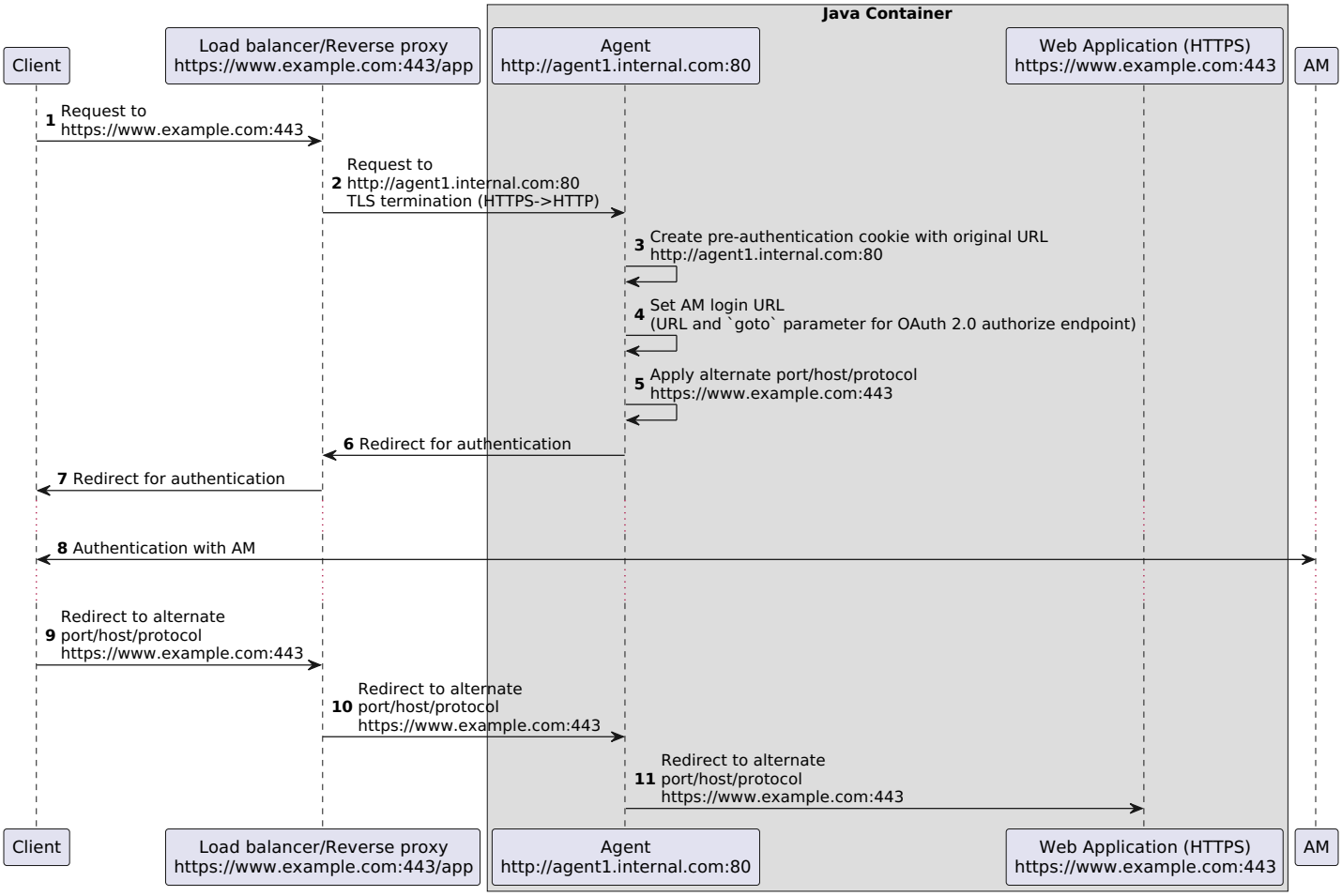
The following diagram shows the agent connected to a client through a reverse proxie and load balancer. The agent connection to the reverse proxy and load balancer is on HTTP and port 80. The client connection is on HTTPS and port 443.



After TLS offloading, the host, port, and protocol of the request is changed to match the request received by the agent; it no longer matches the request from the client, as shown in the following data flow. The agent uses this URL for the redirect_url from the OAuth 2.0 flow, which causes the request to fail.



In the following flow, the agent overrides the host, port, and protocol for subsequent redirects:

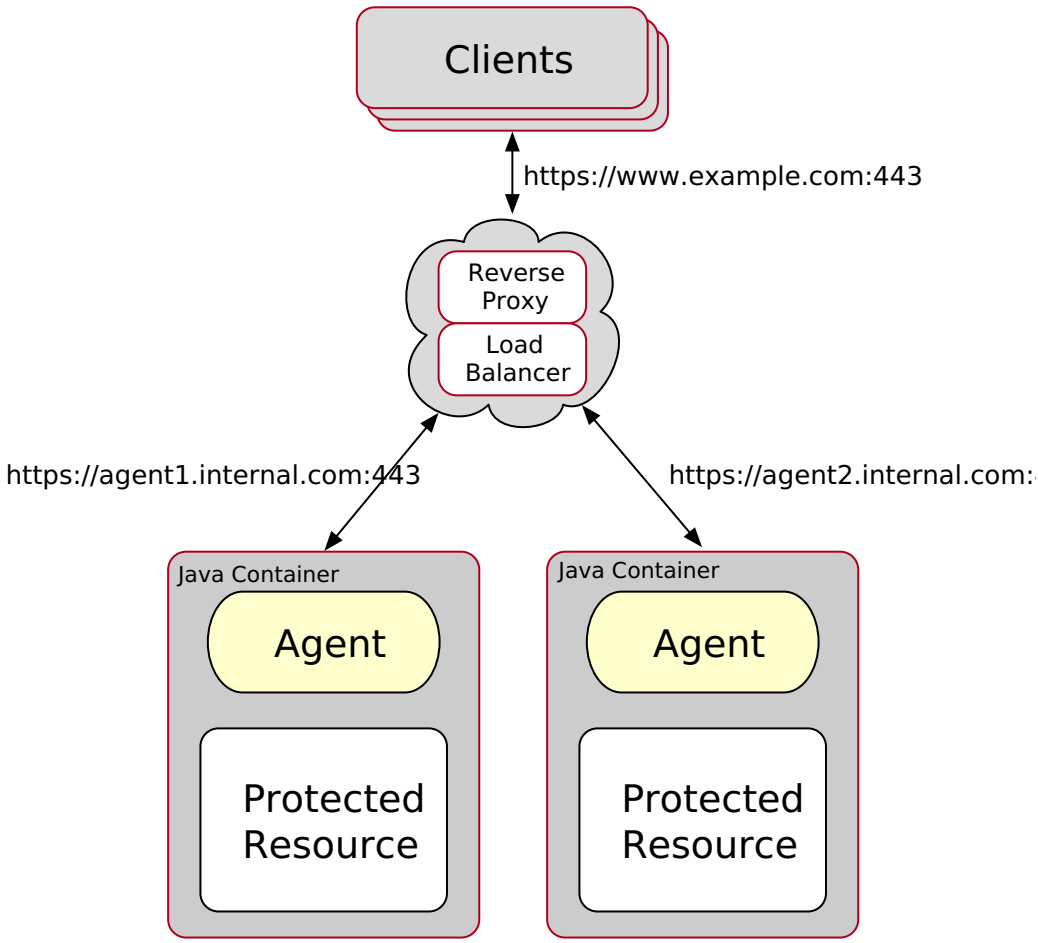


For this scenario, configure the agent as described in To Override Protocol, Host, and Port.

Match FQDNs for request forwarding

The load balancer or reverse proxy forwards requests and responses between clients and protected Java containers only. In this case, ports and protocols configured in the Java container match those on the load balancer or reverse proxy, but FQDNs do not.

The following diagram illustrates this scenario:



For this scenario, configure the agent as described in To Map the Agent Host Name to the Load Balancer or Reverse Proxy Host Name.

Override protocol, host, and port

Use the alternate agent URL properties to override the agent protocol, host, and port with that of the load balancer or reverse proxy.

The agent in this example is in remote configuration mode, but the steps mention properties for agents in local configuration mode.

> **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 and not 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 container directly, from behind the load balancer.

1. Log in to the AM admin UI as an administrative user with rights to modify the agent profile.
2. Select **Realms** > realm name > **Applications** > **Agents** > **Java** > agent name > **Advanced**.
3. Set **Alternative Agent Host Name** to that of the load balancer or reverse proxy. For example, `lb.example.com`.

   The equivalent property setting is `org.forgerock.agents.agent.hostname=lb.example.com`.

4. Set **Alternative Agent Port** to that of the load balancer or proxy. For example, `80`.

   The equivalent property setting is `org.forgerock.agents.agent.port=80`.

5. Set **Alternative Agent Protocol** to that of the load balancer or proxy. For example, `http` or `https`.

   The equivalent property setting is `org.forgerock.agents.agent.protocol=https`.

6. Save your work.

7. Restart the Java container where the agent is installed.

Map agent host name to the load balancer or reverse proxy host name

When protocols and port numbers match, configure FQDN mapping.

The agent in this example is in remote configuration mode, but the steps mention properties for agents in local configuration mode.

1. Log in to the AM admin UI as an administrative user with rights to modify the Java agent profile.

2. Select **Realms** > realm name > **Applications** > **Agents** > **Java** > agent name.

3. In the **Global** tab, enable **FQDN Check**.

   The equivalent property setting is `org.forgerock.agents.fqdn.check.enabled=true`.

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

   The equivalent property setting is `org.forgerock.agents.fqdn.default=lb.example.com`.

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

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

   The equivalent property setting is `org.forgerock.agents.fqdn.map[agent.example.com]=lb.example.com`.

7. Save your work.

8. Restart the Java container where the agent is installed.

Configure 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.

The agent in this example is in remote configuration mode, but the steps mention properties for agents in local configuration mode.

1. Log in to the AM admin UI with a user that has permissions to modify the Java agent profile.

2. Select **Realms** > realm name > **Applications** > **Agents** > **Java** > agent name > **Advanced**.

3. In the Client IP Address Header field, configure the name of the header containing the IP address of the client. For example, `X-Forwarded-For`.

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

   The equivalent property setting is `org.forgerock.agents.http.header.containing.ip.address=X-Forwarded-For`.

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

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

   The equivalent property setting is `org.forgerock.agents.http.header.containing.remote.hostname=X-Forwarded-Host`.

5. Save your changes.

Configure POST data preservation for load balancers or reverse proxies

Configure the load balancer or reverse proxy **and** the agents for session stickiness.

The agent in this example is in remote configuration mode, but the steps mention properties for agents in local configuration mode.

1. Log in to the AM admin UI as a user with permission to modify the agent profile.
2. Select **Realms** > realm name > **Applications** > **Agents** > **Java** > agent name > **Advanced**.
3. Decide whether the agent should create a cookie or append a string to the URL to assist with sticky load balancing.

   In **PDP Sticky session mode**, configure one of the following options:

   - **Cookie**: The agent creates a cookie for POST data preservation session stickiness. The contents 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 `org.forgerock.agents.pdp.sticky.session.mode=Cookie|URL]`.

4. In the POST Data Preservation Sticky Session Key Value property, configure a key-pair value separated by the `=` character.

   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 `org.forgerock.agents.pdp.sticky.session.value=lb=myserver`.

5. Save your changes.
6. Configure your load balancer or reverse proxy to ensure session stickiness when the cookie or URL query parameter are present.

Configure an Apache HTTP Server as a reverse proxy

This section provides an example of how to configure Apache as a reverse proxy between AM and the agent. You can use any reverse proxy that supports the WebSocket protocol.

Refer to the Apache documentation to configure Apache for load balancing and any other requirement for your environment.

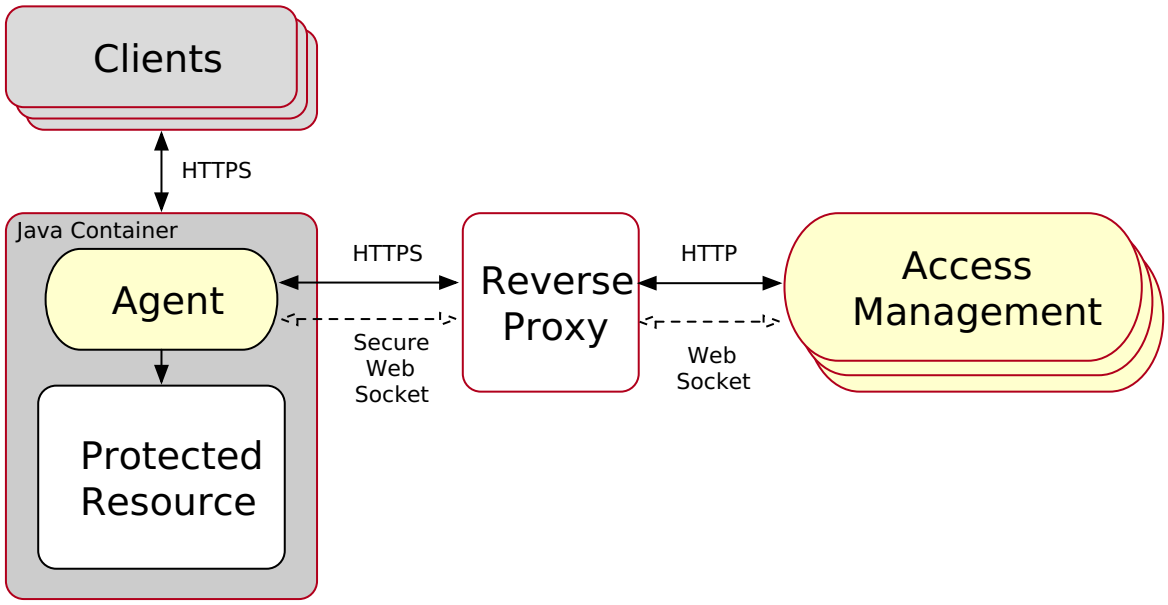


Figure 1. Reverse Proxy Configured Between the Agent and AM

Note that the communication protocol changes from HTTPS to HTTP.

Configure Apache as a Reverse Proxy Example

1. In your deployed reverse proxy instance, locate the `httpd.conf` file.
2. Add the following modules required for a proxy configuration:

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

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

3. Add the proxy configuration inside the `VirtualHost` context, and set the following directives:

```
<VirtualHost 192.168.1.1>
...
# Proxy Config
RequestHeader set X-Forwarded-Proto "https" (1)
ProxyPass "/am/notifications" "ws://am.example.com:8080/am/notifications"
Upgrade=websocket (2)
ProxyPass "/am" "http://am.example.com:8080/am" (3)
ProxyPassReverseCookieDomain "am.internal.example.com" "proxy.example.com" (4)
ProxyPassReverse "/am" "http://am.example.com:8080/am" (5)
...
</VirtualHost>
```

1. **RequestHeader**: If the proxy is configured for https, set to `https`. Otherwise, set to `http`. A later step configures AM to recognize the forwarded header and use it in the `goto` parameter, to redirect back to the Java Agent after authentication.
2. **ProxyPass**: Allow WebSocket traffic between AM and the Java Agent. If HTTPS is configured between the proxy and AM, use `wss` instead of `ws`.
3. **ProxyPass**: Allow HTTP traffic between AM and the agent.
4. **ProxyPassReverseCookieDomain**: Rewrite the domain string of Set-Cookie headers in this format: `internal domain (AM's domain) public domain (proxy's domain)`.
5. **ProxyPassReverse**: Set to the same value configured for the `ProxyPass` directive.

4. Restart the reverse proxy instance.
5. Configure AM to recover the forwarded header 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 Communication Between AM and Agents

Implement a custom task handler

This section describes how to add a custom task handler to the list of handlers, and provides example handlers. At startup, Java Agent tries to instantiate the specified service resolver class. If unsuccessful, it instantiates the original service resolver.

1. Place `com.sun.identity.agents.arch.ServiceResolver` on the classpath.
2. Add `com.sun.identity.agents.arch.ServiceResolver` to the bootstrap property Service Resolver Class Name.

Use the following functions to return a list of class names to customize the task handler:

Function	When to execute the class	What the class must implement
`List<String> getPreInboundTaskHandlers()`	Before all other inbound task handlers	`IAmFilterTaskHandler`
`List<String> getPostInboundTaskHandlers()`	After all other inbound task handlers	`IAmFilterTaskHandler`
`List<String> getPreSelfRedirectHandlers()`	Before all other self-redirect task handlers	`IAmFilterTaskHandler`

Function	When to execute the class	What the class must implement
`List<String> getPostSelfRedirectHandlers()`	After all other self-redirect task handlers	`IAmFilterTaskHandler`
`List<String> getPreFilterResultHandlers()`	Before all other result handlers	`IAmFilterResultHandler`
`List<String> getPostFilterResultHandlers()`	After all other result handlers	`IAmFilterResultHandler`.

If the named handler classes are not on the classpath, or do not implement the required interface, then:

- Handler instantiation fails.
- A message is logged at ERROR level.
- The agent abandons processing and returns an HTTP 500, effectively denying all requests.

When a handler list is built, make sure that any `isActive` function implemented by your custom handler returns `true`, if appropriate. Any handler returning `false` is evicted.

For each InboundTaskHandler and SelfRedirectHandler, the `process` function is invoked until a non-null value, such as `continue` or `block`, is returned. The non-null value becomes the result for that resource access. Returning a null value indicates to carry on to the other handlers.

For FilterResultHandlers, returning a null value causes an error.

Example custom filter result task handler

```
/*
 * Copyright 2019-2024 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with ForgeRock AS.
 * or with one of its affiliates. All use shall be exclusively subject
 * to such license between the licensee and ForgeRock AS.
 */
package com.sun.identity.agents.custom;

import static org.forgerock.agents.debug.AgentDebug.logTrace;

import javax.servlet.http.HttpServletRequest;

import org.forgerock.agents.util.Utils;

import com.sun.identity.agents.arch.AgentConfiguration;
import com.sun.identity.agents.arch.Manager;
import com.sun.identity.agents.filter.AmFilterMode;
import com.sun.identity.agents.filter.AmFilterRequestContext;
import com.sun.identity.agents.filter.AmFilterResult;
import com.sun.identity.agents.filter.AmFilterResultHandler;

/**
 * This is an example of a custom filter result task handler
 */
@SuppressWarnings("unused")
public class CustomFilterResultTaskHandler extends AmFilterResultHandler {

    public CustomFilterResultTaskHandler(Manager manager) {
        super(manager);
    }

    @Override
    public boolean isActive() {
        return true;
    }

    @Override
    public String getHandlerName() {
        return "CustomFilterResultTaskHandler";
    }

    @Override
    public AmFilterResult process(AmFilterRequestContext context, AmFilterResult result) {

        String applicationName = Utils.getApplicationName(context);
        AmFilterMode amFilterMode = AgentConfiguration.getTheFilterMode(applicationName);
        HttpServletRequest request = context.getHttpServletRequest();

        logTrace("Hello from {}, application name {}, filter mode {}, {} {}, result {}",
                getHandlerName(), applicationName, amFilterMode,
                request.getMethod(), request.getRequestURI(),
                result.toString());


        // Must return the result parameter, unless you have a really good reason not to.
        return result;
    }
}
```

Example custom self-redirect task handler

```
/*
 * Copyright 2019-2024 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with ForgeRock AS.
 * or with one of its affiliates. All use shall be exclusively subject
 * to such license between the licensee and ForgeRock AS.
 */
package com.sun.identity.agents.custom;

import static org.forgerock.agents.debug.AgentDebug.logTrace;

import javax.servlet.http.HttpServletRequest;

import org.forgerock.agents.util.Utils;

import com.sun.identity.agents.arch.AgentConfiguration;
import com.sun.identity.agents.arch.AgentException;
import com.sun.identity.agents.arch.Manager;
import com.sun.identity.agents.filter.AmFilterMode;
import com.sun.identity.agents.filter.AmFilterRequestContext;
import com.sun.identity.agents.filter.AmFilterResult;
import com.sun.identity.agents.filter.AmFilterTaskHandler;
import com.sun.identity.agents.filter.IBaseAuthnContext;

/**
 * This is an example of a custom self-redirect task handler.  It is essentially the same as the inbound task
 * handler.
 */
@SuppressWarnings("unused")
public class CustomSelfRedirectTaskHandler extends AmFilterTaskHandler {

    public CustomSelfRedirectTaskHandler(Manager manager) {
        super(manager);
    }

    @Override
    public void initialize(IBaseAuthnContext context) throws AgentException {
        super.initialize(context);
    }

    @Override
    public boolean isActive() {
        return true;
    }

    @Override
    public String getHandlerName() {
        return "Custom self redirect task handler";
    }

    @Override
    public AmFilterResult process(AmFilterRequestContext context) {

        String applicationName = Utils.getApplicationName(context);
        AmFilterMode amFilterMode = AgentConfiguration.getTheFilterMode(applicationName);
        HttpServletRequest request = context.getHttpServletRequest();

        logTrace("Hello from {}, application name {}, filter mode {}, {} {}",
                getHandlerName(), applicationName, amFilterMode,
                request.getMethod(), request.getRequestURI());
```

```
        // return null to continue to the other task handlers (until one returns a non null value)
        // return AmFilterResultStatus.STATUS_CONTINUE to grant access (continue to the next filter after the agent)
        // return AmFilterResultStatus.STATUS_REDIRECT to redirect somewhere else
        // return AmFilterResultStatus.STATUS_FORBIDDEN to deny access
        // return AmFilterResultStatus.STATUS_SERVE_DATA to serve up data to the browser
        // return AmFilterResultStatus.STATUS_SERVER_ERROR to abort the request with a 500 server error
        //
        return null;
    }
}
```

Example custom inbound task handler

```
/*
 * Copyright 2019-2024 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with ForgeRock AS.
 * or with one of its affiliates. All use shall be exclusively subject
 * to such license between the licensee and ForgeRock AS.
 */
package com.sun.identity.agents.custom;

import static org.forgerock.agents.debug.AgentDebug.logTrace;

import javax.servlet.http.HttpServletRequest;

import org.forgerock.agents.util.Utils;

import com.sun.identity.agents.arch.AgentConfiguration;
import com.sun.identity.agents.arch.AgentException;
import com.sun.identity.agents.arch.Manager;
import com.sun.identity.agents.filter.AmFilterMode;
import com.sun.identity.agents.filter.AmFilterRequestContext;
import com.sun.identity.agents.filter.AmFilterResult;
import com.sun.identity.agents.filter.AmFilterTaskHandler;
import com.sun.identity.agents.filter.IBaseAuthnContext;

/**
 * This is an example of a custom inbound task handler
 */
@SuppressWarnings("unused")
public class CustomInboundTaskHandler extends AmFilterTaskHandler {

    public CustomInboundTaskHandler(Manager manager) {
        super(manager);
    }

    @Override
    public void initialize(IBaseAuthnContext context) throws AgentException {
        super.initialize(context);
    }

    @Override
    public boolean isActive() {
        return true;
    }

    @Override
    public String getHandlerName() {
        return "Custom inbound task handler";
    }

    @Override
    public AmFilterResult process(AmFilterRequestContext context) {

        String applicationName = Utils.getApplicationName(context);
        AmFilterMode amFilterMode = AgentConfiguration.getTheFilterMode(applicationName);
        HttpServletRequest request = context.getHttpServletRequest();

        logTrace("Hello from {}, application name {}, filter mode {}, {} {}",
                getHandlerName(), applicationName, amFilterMode,
                request.getMethod(), request.getRequestURI());
```

```
        // return null to continue to the other task handlers (until one returns a non null value)
        // return AmFilterResultStatus.STATUS_CONTINUE to grant access (continue to the next filter after the agent)
        // return AmFilterResultStatus.STATUS_REDIRECT to redirect somewhere else
        // return AmFilterResultStatus.STATUS_FORBIDDEN to deny access
        // return AmFilterResultStatus.STATUS_SERVE_DATA to serve up data to the browser
        // return AmFilterResultStatus.STATUS_SERVER_ERROR to abort the request with a 500 server error
        //
        return null;
    }
}
```

Example of how to override the ServiceResolver class

```
/*
 * Copyright 2019-2024 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with ForgeRock AS.
 * or with one of its affiliates. All use shall be exclusively subject
 * to such license between the licensee and ForgeRock AS.
 */
package com.sun.identity.agents.custom;

import java.util.ArrayList;
import java.util.List;

import com.sun.identity.agents.arch.ServiceResolver;

/**
 * This is an example of how to override the ServiceResolver class to provide your own custom task handlers.  To use
 * this example class, place the following in the custom properties on the advanced tab in the Java Agents profile:
 * <p></p>
 * org.forgerock.agents.service.resolver.class.name=com.sun.identity.agents.custom.CustomServiceResolverExample
 * <p></p>
 * and restart the agent.
 */
@SuppressWarnings("unused")
public class CustomServiceResolverExample extends ServiceResolver {

    @Override
    public List<String> getPreInboundTaskHandlers() {
        List<String> result = new ArrayList<>();
        result.add(CustomInboundTaskHandler.class.getName());
        return result;
    }

    @Override
    public List<String> getPostInboundTaskHandlers() {
        return new ArrayList<>();
    }

    @Override
    public List<String> getPreSelfRedirectHandlers() {
        List<String> result = new ArrayList<>();
        result.add(CustomSelfRedirectTaskHandler.class.getName());
        return result;
    }

    @Override
    public List<String> getPostSelfRedirectHandlers() {
        return new ArrayList<>();
    }

    @Override
    public List<String> getPreFilterResultHandlers() {
        List<String> result = new ArrayList<>();
        result.add(CustomFilterResultTaskHandler.class.getName());
        return result;
    }
```

```
    @Override
    public List<String> getPostFilterResultHandlers() {
        return new ArrayList<>();
    }
}
```

Troubleshooting

ForgeRock provides support services, professional services, training through ForgeRock University, and partner services to help you set up and maintain your deployments. For help, 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 or container and version
    - Java version
    - Patches or other software that might affect the problem

Questions and answers

Question: Why does the agent fails to initialize, with an error like this:

```
Servlet failed with an Exception: java.lang.NoSuchMethodError:
com.sun.identity.shared.configuration.SystemPropertiesManager.getAsInt(Ljava/lang/String;I)I
```

Answer: An out-of-date version of the SystemPropertiesManager class is deployed in another jar, that did not have the `getAsInt` function, and the classloader chose the outdated class instead of the agent's class.

Follow these steps to help locate jars containing outdated classes, by showing where these classes are loaded from:

1. Add the property `-Ddisplay.classpath.mode.enabled=true` to the JVM properties in the container where the agent runs, and restart the container.

   ```
   JAVA_OPTS="$JAVA_OPTS -Ddisplay.classpath.mode.enabled=true"
   ```

2. Access the URL of a protected resource, using the class name as a query parameter. The following example requests the path of the SystemPropertiesManager package:

   ```
   https://www.example.com/myapp/index.html?com.sun.identity.shared.configuration.SystemPropertiesManager
   ```

   If the agent finds the package, it displays the path. Otherwise, it displays an error.

3. After troubleshooting, remove `Ddisplay.classpath.mode.enabled=true` from the JVM properties. While it is present, the agent returns the class path but performs no other function.

Question:

What should I do if I get an error like this when installing an agent with an HTTPS connection to AM:

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

WARNING: Unable to connect to AM server URL....

If the AM server is TLS-enabled and the root CA certificate for the AM
server certificate has been not imported into the installer JVMs key store (see
installer-logs/debug/Agent.log for detailed exception), import the root
CA certificate and restart the installer; or continue installation without
verifying the AM server URL.
```

Answer:

The Java platform includes certificates from many certificate authorities (CAs). If, however, you run your own CA, or you use self-signed certificates for HTTPS on the web application container where you run AM, then the `agentadmin` command cannot trust the certificate presented during connection to AM, and so cannot complete installation correctly.

After setting up the web application container where you run AM to use HTTPS, get the certificate to trust in a certificate file. The certificate you want is that of the CA who signed the container certificate, or the certificate itself if the container certificate is self-signed.

Copy the certificate file to the system where you plan to install the Java agent. Import the certificate into a trust store that you will use during Java agent installation. If you import the certificate into the default trust store for the Java platform, then the `agentadmin` command can recognize it without additional configuration.

For information about export and import of self-signed certificates, refer to Configuring AM's container for HTTPS of AM's *Installation guide*.

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:

```
<timestamp> +0000 ERROR [c53...348]state identifier not present in authentication state
<timestamp> +0000 WARNING [c53...348]unable to verify pre-authentication cookie
<timestamp> +0000 WARNING [c53...348]convert_request_after_authn_post(): unable to retrieve pre-authentication
request data
<timestamp> +0000 DEBUG [c53...348] exit status: forbidden (3), HTTP status: 403, subrequest 0
```

What is happening?

Answer:

This redirection loop happens because the client-based (stateless) session cookie is surpassing the maximum supported browser header size. Because 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 I see the following message in the agent log:

```
redirect_uri_mismatch. The redirection URI provided does not match a pre-registered value.
```

What should I do?

Answer:

Java Agent accept 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, agents prevent you from accessing the agent on URLs not defined in the Agent Root URL for CDSSO property. Add entries to this property when:

- Configuring Alternative Agent Protocol to access the agent through different protocols. For example, `http://agent.example.com/` and `https://agent.example.com/`.
- Configuring Alternative Agent Host Name to access the agent through different virtual host names. For example, `http://agent.example.com/` and `http://internal.example.com/`.
- Configuring Alternative Agent Port Number to access the agent through different ports. For example, `http://agent.example.com/` and `http://agent.example.com:8080/`.

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 filter

A servlet that intercepts inbound client requests to a resource, and processes them according to the value of Agent Filter Mode Map.

Agent profile

A set of configuration properties that define the behavior of the agent.

Agent profile realm

The realm in which the agent profile is stored. See also agent profile.

Application

A service exposing protected resources. See Web Application.

In AM policies, an 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 the following:

- A preset list of actions and functional logic, such as policy lookup and resource comparator logic.
- Internal normalization, indexing logic, and comparator logic for applications.

Attribute-based access control (ABAC)

Access control that is based on attributes of a user, such as how old a user is or whether the user is a paying customer.

Authentication

The act of confirming the identity of a principal.

Authentication chaining

A series of authentication modules configured together which a principal must negotiate as configured in order to authenticate successfully.

Authentication level

Positive integer associated with an authentication module, usually used to require success with more stringent authentication measures when requesting resources requiring special protection.

Authentication module

AM authentication unit that handles one way of obtaining and verifying credentials.

Authorization

The act of determining whether to grant or to deny a principal access to a resource.

Authorization server

In OAuth 2.0, issues access tokens to the client after authenticating a resource owner and confirming that the owner authorizes the client to access the protected resource. AM can play this role in the OAuth 2.0 authorization framework.

Auto-federation

Arrangement to federate a principal's identity automatically based on a common attribute value shared across the principal's profiles at different providers.

Autonomous mode

The agent operates independently of AM, without needing to contact an AM instance. Agents allow access to resources as defined in not-enforced lists; otherwise, they deny access.

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

Replaced by remote configuration mode.

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 web applications involved in SSO when a session ends.

Custom login redirect

A mode to use SSO tokens or OpenID Connect (OIDC) ID tokens as session tokens, and redirect login to any endpoint. For more information, refer to Login redirect.

Default login redirect

A mode to use OpenID Connect (OIDC) ID tokens as session tokens, and redirect login to the `/oauth2/authorize` endpoint in the AM instance specified during installation. For more information, refer to Login redirect.

Delegation

Granting users administrative privileges with AM.

Entitlement

Decision that defines the following:

- Which resource names can and cannot be accessed for a given identity in the context of a particular web application.
- Which actions are allowed and denied.
- Related advice and attributes.

Extended metadata

Federation configuration information specific to AM.

Extensible Access Control Markup Language (XACML)

Standard, XML-based access control policy language, including a processing model for making authorization decisions based on policies.

Federation

Standardized means for aggregating identities, sharing authentication and authorization data information between trusted providers, and allowing principals to access services across different providers without authenticating repeatedly.

Fedlet

Service provider application capable of participating in a circle of trust and allowing federation without installing all of AM on the service provider side; AM lets you create Java Fedlets.

Hot swappable

Refers to configuration properties for which changes can take effect without restarting the container where AM runs.

Identity

Set of data that uniquely describes a person or a thing such as a device or a web application.

Identity federation

Linking of a principal's identity across multiple providers.

Identity provider (IdP)

Entity that produces assertions about a principal (such as how and when a principal authenticated, or that the principal's profile has a specified attribute value).

Identity repository

Data store holding user profiles and group information; different identity repositories can be defined for different realms.

Java Agent

Java web application installed in a web container that acts as a policy enforcement point. The Java Agent filters requests to other applications in the container, using policies based on web application resource URLs.

Local configuration mode

The agent reads its configuration from the `AgentConfiguration.properties` file. See also remote configuration mode.

The configuration mode is defined by Location of Agent Configuration Repository.

Login redirect

Java Agent manages login redirect in the following ways: default login redirect and custom login redirect.

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

Entity that intercepts a request for a resource and then enforces policy decisions from a policy decision point.

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 a web application), and is therefore distinguished from other entities.

When a Subject successfully authenticates, AM associates the Subject with the Principal.

Protected resource

A resource that is not matched by a "not-enforced" rule.

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 web 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.

Remote configuration mode

The agent ignores the configuration in `AgentConfiguration.properties`, retains the retrieved bootstrap properties, and downloads the configuration from AM. See also local configuration mode.

The configuration mode is defined by Location of Agent Configuration Repository.

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).

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 web applications by setting a session cookie. See also CTS-based sessions and Client-based sessions.

Session high availability

Capability that lets any AM server in a clustered deployment access shared, persistent information about users' sessions from the CTS token store. The user does not need to log in again unless the entire deployment goes down.

Session token

Unique identifier issued by AM after successful authentication. For a CTS-based sessions, the session token is used to track a principal's session.

Single log out (SLO)

Capability to end a session once, and thereby end the session across multiple web applications.

Single sign-on (SSO)

Capability to authenticate once and gain access to multiple web 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. See also Client-based sessions and CTS-based sessions.

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 realm in which a user is authenticated.

Identity store

Data storage service holding principals' profiles; underlying storage can be an LDAP directory service or a custom `IdRepo` implementation.

Web Agent

Native library installed in a web server that acts as a policy enforcement point with policies based on web page URLs.

Web application

An application that runs on a web server, that is accessed by the user through a web browser. The web application exposes protected resources.

Maintenance guide

This guide describes how to perform recurring administrative operations in PingAM Java Agent.

About Ping Identity Platform™ software

Ping 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

Java Agent 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.

Java Agent 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.

Learn more about audit logging from Audit logging in AM's *Security guide*.

Locally

Log audit events in JSON format to a file in the agent installation directory, `/java_agents/agent_type/logs/audit/` .

Remotely and locally

Log audit events:

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

The following is an example of an agent log record:

```
{
   "timestamp":"...",
   "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":"Java Policy Agent",
   "realm":"/",
   "server":{
      "ip":"127.0.0.1",
      "port":8020
   },
   "client":{
      "ip":"127.0.0.1",
      "port":55180
   },
   "request":{
      "protocol":"HTTP/1.1",
      "operation":"GET"
   },
   "http":{
      "request":{
         "secure":false,
         "method":"GET",
         "path":"http://my.example.com:8020/examples/",
         "headers":{
            "referer":[
               "https://am.example.com:8443/am/oauth2/authorize?scope[...]"
            ],
            "accept-language":[
               "en,en-US;q=0.8,da;q=0.6,fr;q=0.4"
            ],
            "host":[
               "my.example.com:8020"
            ],
            "upgrade-insecure-requests":[
               "1"
            ],
            "connection":[
               "keep-alive"
            ],
            "cache-control":[
               "max-age=0"
            ],
            "accept-encoding":[
               "gzip, deflate"
            ],
            "user-agent":[
               "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) AppleWebKit/537.36 (KHTML, like Gecko)[...]"
            ],
            "accept":[
               "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8"
            ]
         },
         "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 Ping Identity Platform. For more information about the audit log format, refer to Audit log format in AM's *Security guide*.

Java Agent supports propagation of the transaction ID across the Ping Identity Platform, using the HTTP header `X-ForgeRock-TransactionId`. For more information about configuring the header, refer to Trust transaction headers in AM's *Security guide*.

Configure audit logging

By default, Java Agent 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 the AM admin UI, select **Realms** > realm name > **Applications** > **Agents** > **Java** > 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. To log audit messages locally, enable Enable Local Audit Log Rotation to rotate the audit log files when they reach a maximum size.
5. If you enabled Enable Local Audit Log Rotation, specify the maximum size of the audit log files in Local Audit Log Rotation Size.

Monitoring

The following sections describe how to set up and maintain monitoring in your deployment, to ensure appropriate performance and service availability.

Monitoring the Prometheus endpoint

All ForgeRock products automatically expose a monitoring endpoint where Prometheus can scrape metrics, in a standard Prometheus format.

For information about installing and running Prometheus, refer to the Prometheus documentation.

By default, no special setup or configuration is required to access metrics at this endpoint. The following example queries the Prometheus endpoint for a route.

Tools such as Grafana are available to create customized charts and graphs based on the information collected by Prometheus. For more information on installing and running Grafana, refer to the Grafana website.

Prometheus performance metrics are provided by an endpoint configured in the protected web application's `web.xml` file. The endpoint must be accessible to the Prometheus server that uses the performance data.

When the example in Exposing endpoints for Prometheus and Common REST is configured, the Prometheus endpoint is available at `https://mydomain.example.com/myapp/metrics/prometheus`.

Monitoring the Common REST monitoring endpoint

Common REST performance metrics are provided by an endpoint configured in the protected web application's `web.xml` file. The endpoint must be accessible to the REST client that uses the performance data.

When the example in Exposing endpoints for Prometheus and Common REST is configured, the Common REST performance monitoring endpoint is available at `https://mydomain.example.com/myapp/metrics/crest`.

Exposing endpoints for Prometheus and Common REST

Use the following procedure to expose endpoints for Prometheus or Common REST.

1. For each protected web application that is to expose metrics, edit the web application's `web.xml` file.

   The following Tomcat example exposes a base endpoint named `/metrics`:

   ```
   <servlet>
      <servlet-name>AgentMonitoring</servlet-name>
      <servlet-class>org.forgerock.http.servlet.HttpFrameworkServlet</servlet-class>
      <init-param>
          <param-name>application-loader</param-name>
          <param-value>guice</param-value>
      </init-param>
   </servlet>
   <servlet-mapping>
       <servlet-name>AgentMonitoring</servlet-name>
       <url-pattern>/metrics/*</url-pattern>
   </servlet-mapping>
   ```

   Choose a name for the exposed base endpoint that does not conflict with any of the built-in agent endpoints, for example `/sunwCDSSORedirectURI`.

2. Allow access to the monitoring endpoint that is protected by the agent, in one of the following ways:

   1. Configure Not-Enforced URIs to create a not-enforced URI rule for the base endpoint.

      The following example rule allows access to the metrics base endpoint:

      ```
      */metrics/*
      ```

2. Configure Not-Enforced URIs, Not-Enforced Client IP List, and Not-Enforced Compound Rule Separator to create a compound not-enforced rule for the base endpoint.

   The rule allows access from only the IP addresses of the REST clients or Prometheus server.

   The following example rule allows access to the `/metrics` endpoint for HTTP requests that come from the IP address range from 192.168.1.1 to 192.168.1.3:

   ```
   192.168.1.1-192.168.1.3 | */metrics/*
   ```

   HTTP requests from other IP addresses are not able to access the metrics base endpoint.

3. Create an authorization policy in AM to restrict access to the metrics base endpoint.

   Note that the metric base endpoint does not require login credentials. You can use a policy to ensure that requests to the endpoints are authenticated against the AM instance.

   For more information, refer to Policies in AM's *Authorization guide*.

3. If the monitoring endpoint is protected by AM policies, include the required credentials.

Writing metrics to CSV files

Configure Export Monitoring Metrics to CSV to write metric information to CSV files.

Summary of metric types

Timer fields

Common REST fields

Field	Description
`_id`	Metric ID.
`_type`	Metric type.
`count`	Number of events recorded for this metric.
`total`	Sum of the durations recorded for this metric.
`min`	Minimum duration recorded for this metric.
`max`	Maximum duration recorded for this metric.
`mean`	Average duration recorded for this metric.
`stddev`	Standard deviation of durations recorded for this metric.
`duration_units`	Units used for measuring the durations in the metric.

Field	Description
`p50`	50% of the durations recorded are at or below this value.
`p75`	75% of the durations recorded are at or below this value.
`p95`	95% of the durations recorded are at or below this value.
`p98`	98% of the durations recorded are at or below this value.
`p99`	99% of the durations recorded are at or below this value.
`p999`	99.9% of the durations recorded are at or below this value.
`m1_rate`	One-minute average rate.
`m5_rate`	Five-minute average rate.
`m15_rate`	Fifteen-minute average rate.
`mean_rate`	Average rate.
`rate_units`	Units used for measuring the rate of the metric.

Note

Duration-based values, such as `min`, `max`, and `p50`, are weighted towards newer data. By representing approximately the last five minutes of data, the timers make it easier to see recent changes in behavior, rather than a uniform average of recordings since the server was started.

The following is an example of the `requests.granted.not-enforced` metric from the Common REST endpoint:

```
{
  "_id" : "requests.granted.not-enforced",
  "_type" : "timer",
  "count" : 486,
  "total" : 80.0,
  "min" : 0.0,
  "max" : 1.0,
  "mean" : 0.1905615495053855,
  "stddev" : 0.39274399467782056,
  "duration_units" : "milliseconds",
  "p50" : 0.0,
  "p75" : 0.0,
  "p95" : 1.0,
  "p98" : 1.0,
  "p99" : 1.0,
  "p999" : 1.0,
  "m1_rate" : 0.1819109974890356,
  "m5_rate" : 0.05433445522996721,
  "m15_rate" : 0.03155662103953588,
  "mean_rate" : 0.020858521722211427,
  "rate_units" : "calls/second"
}
```


Prometheus fields

The Prometheus endpoint does not provide rate-based statistics, as rates can be calculated from the time-series data.

Field	Description
`# TYPE`	Metric ID, and type. Note that the `Timer` metric type is reported as a `Summary` type. Formatted as a comment.
`_count`	Number of events recorded.
`_total`	Sum of the durations recorded.
`{quantile="0.5"}`	50% of the durations are at or below this value.
`{quantile="0.75"}`	75% of the durations are at or below this value.
`{quantile="0.95"}`	95% of the durations are at or below this value.
`{quantile="0.98"}`	98% of the durations are at or below this value.
`{quantile="0.99"}`	99% of the durations are at or below this value.
`{quantile="0.999"}`	99.9% of the durations are at or below this value.

> **Note**
>
> Duration-based quantile values are weighted towards newer data. By representing approximately the last five minutes of data, the timers make it easier to see recent changes in behavior, rather than a uniform average of recordings since the server was started.

The following is an example of the `ja_requests{access=granted,decision=allowed-by-policy}` metric from the Prometheus endpoint:

```
ja_requests_seconds{access="granted",decision="allowed-by-policy",quantile="0.5",} 0.013000000000000001
ja_requests_seconds{access="granted",decision="allowed-by-policy",quantile="0.75",} 0.022000000000000002
ja_requests_seconds{access="granted",decision="allowed-by-policy",quantile="0.95",} 0.022000000000000002
ja_requests_seconds{access="granted",decision="allowed-by-policy",quantile="0.98",} 0.022000000000000002
ja_requests_seconds{access="granted",decision="allowed-by-policy",quantile="0.99",} 0.022000000000000002
ja_requests_seconds{access="granted",decision="allowed-by-policy",quantile="0.999",} 1.1380000000000001
ja_requests_count{access="granted",decision="allowed-by-policy",} 7.0
ja_requests_seconds_total{access="granted",decision="allowed-by-policy",} 1.21
```

Gauge fields

Common REST fields

Metric for a numerical value that can increase or decrease. The value for a gauge is calculated when requested, and represents the state of Metric at that specific time.

Field	Description
`_id`	Metric ID.
`_type`	Metric type.
`value`	Current value of the metric.

The following is an example of the `jvm.used-memory` metric from the Common REST endpoint:

```
{
  "_id" : "jvm.used-memory",
  "_type" : "gauge",
  "value" : 2.13385216E9
}
```

Prometheus fields

Field	Description
`# TYPE`	Metric ID, and type. Formatted as a comment.
`{Metric ID}`	Current value. Large values may be represented in scientific E-notation.

The following is an example of the `ja_jvm_used_memory_bytes` metric from the Prometheus endpoint:

```
# TYPE ja_jvm_used_memory_bytes gauge
ja_jvm_used_memory_bytes 1.418723328E9
```

Distinct counter

Metric providing an estimate of the number of *unique* values recorded.

For example, this could be used to estimate the number of unique users who have authenticated, or unique client IP addresses.

> **Note**
>
> The `DistinctCounter` metric is calculated per instance of AM, and cannot be aggregated across multiple instances to get a site-wide view.

Common REST fields

Field	Description
`_id`	Metric ID.
`_type`	Metric type. Note that the `distinctCounter` type is reported as a `gauge` type. The output formats are identical.
`value`	Calculated estimate of the number of unique values recorded in the metric.

The following is an example of the `authentication.unique-uuid.success` metric from the Common REST endpoint:

```
{
  "_id" : "authentication.unique-uuid.success",
  "_type" : "gauge",
  "value" : 3.0
}
```

Prometheus fields

Field	Description
`# TYPE`	Metric ID, and type. Note that the `distinctCounter` type is reported as a `gauge` type. The output formats are identical. Formatted as a comment.
`{Metric ID}`	Calculated estimate of the number of unique values recorded in the metric.

The following is an example of the `ja_notenforced_ip_unmatched_cache_size` metric from the Prometheus endpoint:

```
# TYPE ja_notenforced_ip_unmatched_cache_size gauge
ja_notenforced_ip_unmatched_cache_size 3.0
```

Summary of exposed metrics

Java Agent exposes the monitoring metrics described in this section.

Audit handler metrics

Metric	Prometheus name	Description
`audit.access.generate`	`ja_audit_generate{topic=access}`	Time taken to generate an audit object. (Timer)
`audit.handler.<handler-type>.default.access.<outcome>`	`ja_audit{handler-type=<handler-type>,name=default,topic=access,outcome=<outcome>}`	Time taken to audit outcomes, both locally to the agent and remotely in AM. (Timer)

Labels:

`<handler-type>`

`am-delegate`. Remote auditing performed by AM. (Prometheus: `am_delegate`)

`json`. Local audit logging using JSON.

`<outcome>`

`success`

`failure`

Endpoint and REST SDK metrics

Metric	Prometheus name	Description
`session-info`	`ja_session_info`	Time taken to retrieve user session information from AM. (Timer)
`user-profile`	`ja_user_profile`	Time taken to retrieve the user profile information from AM. (Timer)
`policy-decision`	`ja_policy_decision`	Time taken to retrieve policy decisions from AM. (Timer)

JSON Web Token (JWT) metrics

Metric	Prometheus name	Description
`jwt.cache.size`	`ja_jwt_cache_size`	Size of the JWT cache. (Gauge)
`jwt.cache.eviction`	`ja_jwt_cache_eviction`	The eviction count for the JWT cache. (Gauge)
`jwt.cache.load-count`	`ja_jwt_cache_load_count`	The load count for the JWT cache. (Gauge)
`jwt.cache.load-time`	`ja_jwt_cache_load_time`	The load time for the JWT cache, in milliseconds. (Gauge)
`jwt.cache.hit`	`ja_jwt_cache{outcome=hit}`	The hit count for the JWT cache. (Gauge)
`jwt.cache.miss`	`ja_jwt_cache{outcome=miss}`	The miss count for the JWT cache. (Gauge)

JVM metrics

Tip

To get Metric name used by Prometheus, prepend `ja_` to the names below, and replace period (`.`) and hyphen (`-`) characters with underscore (`_`) characters. For example, the `jvm.available-cpus` metric is named `ja_jvm_available_cpus` in Prometheus.

Name	Description
`jvm.available-cpus`	Number of processors available to the Java virtual machine. (Gauge)
`jvm.class-loading.loaded`	Number of classes loaded since the Java virtual machine started. (Gauge)
`jvm.class-loading.unloaded`	Number of classes unloaded since the Java virtual machine started. (Gauge)
`jvm.garbage-collector.PS-MarkSweep.count`	Number of collections performed by the "parallel scavenge mark sweep" garbage collection algorithm. (Gauge)
`jvm.garbage-collector.PS-MarkSweep.time`	Approximate accumulated time taken by the "parallel scavenge mark sweep" garbage collection algorithm. (Gauge)
`jvm.garbage-collector.PS-Scavenge.count`	Number of collections performed by the "parallel scavenge" garbage collection algorithm. (Gauge)

Name	Description
`jvm.garbage-collector.PS-Scavenge.time`	Approximate accumulated time taken by the "parallel scavenge" garbage collection algorithm. (Gauge)
`jvm.memory-usage.heap.init`	Amount of heap memory that the Java virtual machine initially requested from the operating system. (Gauge)
`jvm.memory-usage.heap.max`	Maximum amount of heap memory that the Java virtual machine will attempt to use. (Gauge)
`jvm.memory-usage.heap.committed`	Amount of heap memory that is committed for the Java virtual machine to use. (Gauge)
`jvm.memory-usage.heap.used`	Amount of heap memory used by the Java virtual machine. (Gauge)
`jvm.memory-usage.total.init`	Amount of memory that the Java virtual machine initially requested from the operating system. (Gauge)
`jvm.memory-usage.total.max`	Maximum amount of memory that the Java virtual machine will attempt to use. (Gauge)
`jvm.memory-usage.non-heap.init`	Amount of non-heap memory that the Java virtual machine initially requested from the operating system. (Gauge)
`jvm.memory-usage.non-heap.max`	Maximum amount of non-heap memory that the Java virtual machine will attempt to use. (Gauge)
`jvm.memory-usage.non-heap.committed`	Amount of non-heap memory that is committed for the Java virtual machine to use. (Gauge)
`jvm.memory-usage.non-heap.used`	Amount of non-heap memory used by the Java virtual machine. (Gauge)
`jvm.memory-usage.pools.Code-Cache.init`	Amount of "code cache" memory that the Java virtual machine initially requested from the operating system. (Gauge)
`jvm.memory-usage.pools.Code-Cache.max`	Maximum amount of "code cache" memory that the Java virtual machine will attempt to use. (Gauge)
`jvm.memory-usage.pools.Code-Cache.committed`	Amount of "code cache" memory that is committed for the Java virtual machine to use. (Gauge)
`jvm.memory-usage.pools.Code-Cache.used`	Amount of "code cache" memory used by the Java virtual machine. (Gauge)

Name	Description
`jvm.memory-usage.pools.Compressed-Class-Space.init`	Amount of "compressed class space" memory that the Java virtual machine initially requested from the operating system. (Gauge)
`jvm.memory-usage.pools.Compressed-Class-Space.init`	Maximum amount of "compressed class space" memory that the Java virtual machine will attempt to use. (Gauge)
`jvm.memory-usage.pools.Compressed-Class-Space.committed`	Amount of "compressed class space" memory that is committed for the Java virtual machine to use. (Gauge)
`jvm.memory-usage.pools.Compressed-Class-Space.used`	Amount of "compressed class space" memory used by the Java virtual machine. (Gauge)
`jvm.memory-usage.pools.Metaspace.init`	Amount of "metaspace" memory that the Java virtual machine initially requested from the operating system. (Gauge)
`jvm.memory-usage.pools.Metaspace.max`	Maximum amount of "metaspace" memory that the Java virtual machine will attempt to use. (Gauge)
`jvm.memory-usage.pools.Metaspace.committed`	Amount of "metaspace" memory that is committed for the Java virtual machine to use. (Gauge)
`jvm.memory-usage.pools.Metaspace.used`	Amount of "metaspace" memory used by the Java virtual machine. (Gauge)
`jvm.memory-usage.pools.PS-Eden-Space.init`	Amount of "parallel scavenge eden space" memory that the Java virtual machine initially requested from the operating system. (Gauge)
`jvm.memory-usage.pools.PS-Eden-Space.max`	Maximum amount of "parallel scavenge eden space" memory that the Java virtual machine will attempt to use. (Gauge)
`jvm.memory-usage.pools.PS-Eden-Space.committed`	Amount of "parallel scavenge eden space" memory that is committed for the Java virtual machine to use. (Gauge)
`jvm.memory-usage.pools.PS-Eden-Space.used-after-gc`	Amount of "parallel scavenge eden space" memory after the last time garbage collection recycled unused objects in this memory pool. (Gauge)
`jvm.memory-usage.pools.PS-Eden-Space.used`	Amount of "parallel scavenge eden space" memory used by the Java virtual machine. (Gauge)
`jvm.memory-usage.pools.PS-Old-Gen.init`	Amount of "parallel scavenge old generation" memory that the Java virtual machine initially requested from the operating system. (Gauge)

Name	Description
`jvm.memory-usage.pools.PS-Old-Gen.max`	Maximum amount of "parallel scavenge old generation" memory that the Java virtual machine will attempt to use. (Gauge)
`jvm.memory-usage.pools.PS-Old-Gen.committed`	Amount of "parallel scavenge old generation" memory that is committed for the Java virtual machine to use. (Gauge)
`jvm.memory-usage.pools.PS-Old-Gen.used-after-gc`	Amount of "parallel scavenge old generation" memory after the last time garbage collection recycled unused objects in this memory pool. (Gauge)
`jvm.memory-usage.pools.PS-Old-Gen.used`	Amount of "parallel scavenge old generation" memory used by the Java virtual machine. (Gauge)
`jvm.memory-usage.pools.PS-Survivor-Space.init`	Amount of "parallel scavenge survivor space" memory that the Java virtual machine initially requested from the operating system. (Gauge)
`jvm.memory-usage.pools.PS-Survivor-Space.max`	Maximum amount of "parallel scavenge survivor space" memory that the Java virtual machine will attempt to use. (Gauge)
`jvm.memory-usage.pools.PS-Survivor-Space.committed`	Amount of "parallel scavenge survivor space" memory that is committed for the Java virtual machine to use. (Gauge)
`jvm.memory-usage.pools.PS-Survivor-Space.used-after-gc`	Amount of "parallel scavenge survivor space" memory after the last time garbage collection recycled unused objects in this memory pool. (Gauge)
`jvm.memory-usage.pools.PS-Survivor-Space.used`	Amount of "parallel scavenge survivor space" memory used by the Java virtual machine. (Gauge)
`jvm.memory-usage.total.committed`	Amount of memory that is committed for the Java virtual machine to use. (Gauge)
`jvm.memory-usage.total.used`	Amount of memory used by the Java virtual machine. (Gauge)
`jvm.thread-state.blocked.count`	Number of threads in the BLOCKED state. (Gauge)
`jvm.thread-state.count`	Number of live threads including both daemon and non-daemon threads. (Gauge)
`jvm.thread-state.daemon.count`	Number of live daemon threads. (Gauge)
`jvm.thread-state.new.count`	Number of threads in the NEW state. (Gauge)
`jvm.thread-state.runnable.count`	Number of threads in the RUNNABLE state. (Gauge)

Name	Description
`jvm.thread-state.terminated.count`	Number of threads in the TERMINATED state. (Gauge)
`jvm.thread-state.timed_waiting.count`	Number of threads in the TIMED_WAITING state. (Gauge)
`jvm.thread-state.waiting.count`	Number of threads in the WAITING state. (Gauge)

Not-enforced rule metrics

Metric	Prometheus name	Description
`notenforced-uri.matched.cache.size`	`ja_notenforced_uri_matched_cache_size`	Size of the not-enforced URI matched cache. (Gauge)
`notenforced-uri.matched.cache.eviction`	`ja_notenforced_uri_matched_cache_eviction`	Eviction count for the not-enforced URI matched cache. (Gauge)
`notenforced-uri.matched.cache.load-count`	`ja_notenforced_uri_matched_cache_load_count`	Load count for the not-enforced URI matched cache. (Gauge)
`notenforced-uri.matched.cache.load-time`	`ja_notenforced_uri_matched_cache_load_time`	Load time for the not-enforced URI matched cache, in milliseconds. (Gauge)
`notenforced-uri.matched.cache.hit`	`ja_notenforced_uri_matched_cache{outcome=hit}`	Hit count for the not-enforced URI matched cache. (Gauge)
`notenforced-uri.matched.cache.miss`	`ja_notenforced_uri_matched_cache{outcome=miss}`	Miss count for the not-enforced URI matched cache. (Gauge)
`notenforced-uri.unmatched.cache.size`	`ja_notenforced_uri_unmatched_cache_size`	Size of the not-enforced URI unmatched cache. (Gauge)
`notenforced-uri.unmatched.cache.eviction`	`ja_notenforced_uri_unmatched_cache_eviction`	Eviction count for the not-enforced URI unmatched cache. (Gauge)
`notenforced-uri.unmatched.cache.load-count`	`ja_notenforced_uri_unmatched_cache_load_count`	Load count for the not-enforced URI unmatched cache. (Gauge)
`notenforced-uri.unmatched.cache.load-time`	`ja_notenforced_uri_unmatched_cache_load_time`	Load time for the not-enforced URI unmatched cache, in milliseconds. (Gauge)
`notenforced-uri.unmatched.cache.hit`	`ja_notenforced_uri_unmatched_cache{outcome=hit}`	Hit count for the not-enforced URI unmatched cache. (Gauge)
`notenforced-uri.unmatched.cache.miss`	`ja_notenforced_uri_unmatched_cache{outcome=miss}`	Miss count for the not-enforced URI unmatched cache. (Gauge)

Metric	Prometheus name	Description
`notenforced-ip.matched.cache.size`	`ja_notenforced_ip_matched_cache_size`	Size of the not-enforced IP matched cache. (Gauge)
`notenforced-ip.matched.cache.eviction`	`ja_notenforced_ip_matched_cache_eviction`	Eviction count for the not-enforced IP matched cache. (Gauge)
`notenforced-ip.matched.cache.load-count`	`ja_notenforced_ip_matched_cache_load_count`	Load count for the not-enforced IP matched cache. (Gauge)
`notenforced-ip.matched.cache.load-time`	`ja_notenforced_ip_matched_cache_load_time`	Load time for the not-enforced IP matched cache, in milliseconds. (Gauge)
`notenforced-ip.matched.cache.hit`	`ja_notenforced_ip_matched_cache{outcome=hit}`	Hit count for the not-enforced IP matched cache. (Gauge)
`notenforced-ip.matched.cache.miss`	`ja_notenforced_ip_matched_cache{outcome=miss}`	Miss count for the not-enforced IP matched cache. (Gauge)
`notenforced-ip.unmatched.cache.size`	`ja_notenforced_ip_unmatched_cache_size`	Size of the not-enforced IP unmatched cache. (Gauge)
`notenforced-ip.unmatched.cache.eviction`	`ja_notenforced_ip_unmatched_cache_eviction`	Eviction count for the not-enforced IP unmatched cache. (Gauge)
`notenforced-ip.unmatched.cache.load-count`	`ja_notenforced_ip_unmatched_cache_load_count`	Load count for the not-enforced IP unmatched cache. (Gauge)
`notenforced-ip.unmatched.cache.load-time`	`ja_notenforced_ip_unmatched_cache_load_time`	Load time for the not-enforced IP unmatched cache, in milliseconds. (Gauge)
`notenforced-ip.unmatched.cache.hit`	`ja_notenforced_ip_unmatched_cache{outcome=hit}`	Hit count for the not-enforced IP unmatched cache. (Gauge)
`notenforced-ip.unmatched.cache.miss`	`ja_notenforced_ip_unmatched_cache{outcome=miss}`	Miss count for the not-enforced IP unmatched cache. (Gauge)
`notenforced-compound.matched.cache.size`	`ja_notenforced_compound_matched_cache_size`	Size of the not-enforced compound matched cache. (Gauge)
`notenforced-compound.matched.cache.eviction`	`ja_notenforced_compound_matched_cache_eviction`	Eviction count for the not-enforced compound matched cache. (Gauge)
`notenforced-compound.matched.cache.load-count`	`ja_notenforced_compound_matched_cache_load_count`	Load count for the not-enforced compound matched cache. (Gauge)

Metric	Prometheus name	Description
`notenforced-compound.matched.cache.load-time`	`ja_notenforced_compound_matched_cache_load_time`	Load time for the not-enforced compound matched cache, in milliseconds. (Gauge)
`notenforced-compound.matched.cache.hit`	`ja_notenforced_compound_matched_cache{outcome=hit}`	Hit count for the not-enforced compound matched cache. (Gauge)
`notenforced-compound.matched.cache.miss`	`ja_notenforced_compound_matched_cache{outcome=miss}`	Miss count for the not-enforced compound matched cache. (Gauge)
`notenforced-compound.unmatched.cache.size`	`ja_notenforced_compound_unmatched_cache_size`	Size of the not-enforced compound unmatched cache. (Gauge)
`notenforced-compound.unmatched.cache.eviction`	`ja_notenforced_compound_unmatched_cache_eviction`	Eviction count for the not-enforced compound unmatched cache. (Gauge)
`notenforced-compound.unmatched.cache.load-count`	`ja_notenforced_compound_unmatched_cache_load_count`	Load count for the not-enforced compound unmatched cache. (Gauge)
`notenforced-compound.unmatched.cache.load-time`	`ja_notenforced_compound_unmatched_cache_load_time`	Load time for the not-enforced compound unmatched cache, in milliseconds. (Gauge)
`notenforced-compound.unmatched.cache.hit`	`ja_notenforced_compound_unmatched_cache{outcome=hit}`	Hit count for the not-enforced compound unmatched cache. (Gauge)
`notenforced-compound.unmatched.cache.miss`	`ja_notenforced_compound_unmatched_cache{outcome=miss}`	Miss count for the not-enforced compound unmatched cache. (Gauge)

Policy decision metrics

Metric	Prometheus name	Description
`policy-decision.cache.size`	`ja_policy_decision_cache_size`	Size of the policy decision cache. (Gauge)
`policy-decision.cache.eviction`	`ja_policy_decision_cache_eviction`	Eviction count for the policy decision cache. (Gauge)
`policy-decision.cache.load-count`	`ja_policy_decision_cache_load_count`	Load count for the policy decision cache. (Gauge)
`policy-decision.cache.load-time`	`ja_policy_decision_cache_load_time`	Load time for the policy decision cache, in milliseconds. (Gauge)

Metric	Prometheus name	Description
`policy-decision.cache.hit`	`ja_policy_decision_cache{outcome=hit}`	Hit count for the policy decision cache. (Gauge)
`policy-decision.cache.miss`	`ja_policy_decision_cache{outcome=miss}`	Miss count for the policy decision cache. (Gauge)

POST data preservation metrics

Metric	Prometheus name	Description
`pdp.cache.size`	`ja_pdp_cache_size`	Size of the POST data preservation cache. (Gauge)
`pdp.cache.eviction`	`ja_pdp_cache_eviction`	Eviction count for the POST data preservation cache. (Gauge)
`pdp.cache.load-count`	`ja_pdp_cache_load_count`	Load count for the POST data preservation cache. (Gauge)
`pdp.cache.load-time`	`ja_pdp_cache_load_time`	Load time for the POST data preservation cache, in milliseconds. (Gauge)
`pdp.cache.hit`	`ja_pdp_cache{outcome=hit}`	Hit count for the POST data preservation cache. (Gauge)
`pdp.cache.miss`	`ja_pdp_cache{outcome=miss}`	Miss count for the POST data preservation cache. (Gauge)

Request metrics

Metric	Prometheus name	Description
`requests.<access>.<decision>`	`ja_requests{access=<access>,decision=<decision>}`	Rate of granted/denied requests and their decision. (Timer)

Labels:

`<access>`

- `granted`
- `denied`

`<decision>`

- `not-enforced` : Request matched a not-enforced rule.

`no-valid-token` : Request did not have a valid SSO token or an OpenID Connect JWT.

`allowed-by-policy` : Request matched a policy, which allowed access.

`denied-by-policy` : Request matched a policy, which denied access.

`am-unavailable` : The AM instance was not reachable.

`agent-exception` : An internal error (exception) occurred within the agent.

Session information metrics

Metric	Prometheus name	Description
`session-info.cache.size`	`ja_session_info_cache_size`	Size of the session information cache. (Gauge)
`session-info.cache.eviction`	`ja_session_info_cache_eviction`	Eviction count for the session information cache. (Gauge)
`session-info.cache.load-count`	`ja_session_info_cache_load_count`	Load count for the session information cache. (Gauge)
`session-info.cache.load-time`	`ja_session_info_cache_load_time`	Load time for the session information cache, in milliseconds. (Gauge)
`session-info.cache.hit`	`ja_session_info_cache{outcome=hit}`	Hit count for the session information cache. (Gauge)
`session-info.cache.miss`	`ja_session_info_cache{outcome=miss}`	Miss count for the session information cache. (Gauge)

SSO token to JWT exchange metrics

Metric	Prometheus name	Description
`sso-exchange.cache.size`	`ja_sso_exchange_cache_size`	Size of the SSO token exchange cache. (Gauge)
`sso-exchange.cache.eviction`	`ja_sso_exchange_cache_eviction`	Eviction count for the SSO token exchange cache. (Gauge)
`sso-exchange.cache.load-count`	`ja_sso_exchange_cache_load_count`	Load count for the SSO token exchange cache. (Gauge)
`sso-exchange.cache.load-time`	`ja_sso_exchange_cache_load_time`	Load time for the SSO token exchange, in milliseconds. (Gauge)
`sso-exchange.cache.hit`	`ja_sso_exchange_cache{outcome=hit}`	Hit count for the SSO token exchange cache. (Gauge)

Metric	Prometheus name	Description
`sso-exchange.cache.miss`	`ja_sso_exchange_cache{outcome=miss}`	Miss count for the SSO token exchange cache. (Gauge)

Websocket metrics

Metric	Prometheus name	Description
`websocket.last-received`	`ja_websocket_last_received`	Number of milliseconds since anything was received over the websocket, for example a ping or a notification. (Gauge)
`websocket.last-sent`	`ja_websocket_last_sent`	Number of milliseconds since anything was sent over the websocket. (Gauge)
`websocket.config-change.received`	`ja_websocket_config_change_received`	Number of configuration change notifications received. Note that some may be ignored if the realm or agent name are not applicable. (DistinctCounter)
`websocket.config-change.processed`	`ja_websocket_config_change_processed`	Number of configuration change notifications processed, that were not ignored. (DistinctCounter)
`websocket.policy-change.received`	`ja_websocket_policy_change_received`	Number of policy change notifications received. Note that some may be ignored if the realm or agent name are not applicable. (DistinctCounter)
`websocket.policy-change.processed`	`ja_websocket_policy_change_processed`	Number of policy change notifications processed, that were not ignored. (DistinctCounter)
`websocket.session-logout.received`	`ja_websocket_session_logout_received`	Number of session logout notifications received. Note that some may be ignored if the realm or agent name are not applicable. (DistinctCounter)
`websocket.session-logout.processed`	`ja_websocket_session_logout_processed`	Number of session logout notifications processed, that were not ignored. (DistinctCounter)
`websocket.ping-pong`	`ja_websocket_ping_pong`	Ping/pong round trip time. (Timer)

Logging

Logs in Java Agent and third-party dependencies are recorded using the Logback implementation of the Simple Logging Facade for Java (SLF4J) API.

By default, the logback configuration is stored in `/path/to/java_agents/agent_type/Agent_n/config/agent-logback.xml`. For information about how to change the location, refer to Agent configuration.

The agent supports the following log levels:

- `TRACE`
- `INFO`
- `WARN`
- `ERROR`
- `ON` (deprecated, this setting uses the `TRACE` log level)
- `OFF` (deprecated, this setting uses the `ERROR` log level)

For a full description of logging options, refer to the Logback website.

Setting the log level

The agent maintains the **last** log level it reads from either `agent-logback.xml` or Agent Debug Level.

> **Tip**
>
> To debug code during the property-reading phase of startup, set the log level in `agent-logback.xml` to `TRACE`.

Remote configuration mode

In remote configuration mode, the agent sets the log level as follows:

- At start up, the agent reads `agent-logback.xml`, and uses the configured log level.
- When the agent receives the first request, it reads `AgentBootstrap.properties`. If org.forgerock.agents.debug.level specifies a different log level, the agent changes the log level.
- The agent then reads the AM properties. If Agent Debug Level specifies different log level, the agent changes the log level.
- While the agent is running, if either of the following events occurs the agent changes the log level to the value configured by the event:
  - Agent Debug Level is updated in AM. The log level is changed immediately.
  - `agent-logback.xml` is updated in the agent, and the update is taken into account as described in Changing `agent-logback.xml`.

Local configuration mode

In local configuration mode, the agent sets the log level as follows:

- At start up, the agent reads `agent-logback.xml`, and uses the configured log level.
- When the agent receives the first request, it reads `AgentBootstrap.properties` and then `AgentConfiguration.properties`.
  - If either file specifies a log level in org.forgerock.agents.debug.level, the agent uses that log level.
  - If both files specify a log level, the agent uses the **last** log level it reads, from `AgentConfiguration.properties`.
- While the agent is running, if either of the following events occurs the agent changes the log level to the value configured by the event:
  - The property org.forgerock.agents.debug.level is changed in `AgentConfiguration.properties`, and the time configured by the property org.forgerock.agents.config.reload.seconds elapses.
  - `agent-logback.xml` is updated in the agent, and the update is taken into account as described in Changing `agent-logback.xml`.

Default logging in `agent-logback.xml`

By default, `agent-logback.xml` is configured as follows:

- When the agent starts, log messages are recorded for the agent.
- Log messages are written to a text file in `/path/to/java_agents/agent_type/Agent_n/logs/debug`, where `/path/to/java_agents/agent_type/Agent_n` is the installation directory.
- The log level is `ERROR`.
- Logs are rolled over once a day, or more frequently if the file reaches 100 MB.

All aspects of the logs are defined by the following reference `agent-logback.xml` deployed by the agent installer:

```
<configuration debug="true" scan="true" scanPeriod="60 seconds">
    <appender name="ROLLING" class="org.forgerock.agents.shaded.ch.qos.logback.core.rolling.RollingFileAppender">

        <file>/full/path/to/logs/debug/directory/log.txt</file>
        <rollingPolicy class="org.forgerock.agents.shaded.ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">

            <!-- rollover daily -->
            <fileNamePattern>log-%d{yyyy-MM-dd}.%i.txt</fileNamePattern>

            <!-- each file should be at most 100MB, keep 30 days worth of history, but at most 10GB -->
            <maxFileSize>100MB</maxFileSize>
            <maxHistory>30</maxHistory>
            <totalSizeCap>10GB</totalSizeCap>
        </rollingPolicy>
        <encoder>
            <pattern>%level %msg%n</pattern>
        </encoder>
    </appender>

    <root level="ERROR">
        <appender-ref ref="ROLLING" />
    </root>
</configuration>
```

Changing `agent-logback.xml`

To change the logging behavior, update `agent-logback.xml`.

To take the change into account, stop and restart the agent, or include a scan and an interval configuration in `agent-logback.xml`. In the following example, `agent-logback.xml` is scanned every 60 seconds:

```
<configuration scan="true" scanPeriod="60 seconds" ... >
  ...
```

For a full description of the options for logging, refer to the Logback website.

Use shaded Logback classnames, to ensure that the agent uses a unique classname compared to other deployed filters and web applications in the container. For example, use

```
org.forgerock.agents.shaded.ch.qos.logback.core.rolling.RollingFileAppender
```

instead of

```
ch.qos.logback.core.rolling.RollingFileAppender
```

Managing errors in `agent-logback.xml`

If `agent-logback.xml` is removed and not replaced, or if a replacement `agent-logback.xml` contains errors, logs are written to the standard output for the container. For example, Tomcat logs are written to `catalina.out`.

To log an error when `agent-logback.xml` fails to load, include a debug configuration:

```
<configuration debug="true" ... >
  ...
```

When `agent-logback.xml` contains errors, messages like these written to the standard output for the container.

```
ERROR in ch.qos.logback.core.joran.spi.Interpreter@20:72 …
ERROR in ch.qos.logback.core.joran.action.AppenderRefAction …
```

Changing the log level in agent-logback.xml

The global log level is set by default to `ERROR` by the following line of the default `agent-logback.xml`:

```
<root level="ERROR">
```

To change the global log level, configure the same line with another log level and take the changed file into account, as described in Changing `agent-logback.xml`.

To change the log level for a single object type without changing it for the rest of the configuration, add a logger defined by the shaded classname of the object, and set its log level.

The following line in `agent-logback.xml` changes the log level for evaluating not-enforced rules, but does not change the log level of other classes or packages:

```
<logger name="org.forgerock.agents.shaded.com.sun.identity.agents.common.NotEnforcedRuleHelper" level="TRACE" />
```

Changing the character set and format of log messages

By default, logs use the system default character set where the agent is running. To use a different character set, or change the pattern of the log messages, edit `agent-logback.xml` to change the `encoder` configuration.

The following lines add the date to log messages, and change the character set:

```
<encoder>
   <pattern>%d{yyyyMMdd-HH:mm:ss} | %-5level | %thread | %logger{20} | %message%n%xException</pattern>
   <charset>UTF-8</charset>
</encoder>
```

Limiting repeated log messages

To keep log files clean and readable, and to prevent log overflow attacks, limit the number of repeat log messages. Add a custom `agent-logback.xml` with a `DuplicateMessageFilter`. This filter detects duplicate messages, and after the specified number of repetitions, drops repeated messages.

The following example allows 5 repetitions of a log message, holds the following 10 repeated messages in the cache, and then discards subsequent repeated messages:

```
<turboFilter class="org.forgerock.agents.shaded.ch.qos.logback.classic.turbo.DuplicateMessageFilter"
allowedRepetitions="5" CacheSize="10" />
```

The DuplicateMessageFilter has the following limitations:

- Filters out **all** duplicate messages. It does not filter per logger, or logger instance, or logger name.
- Detects repetition of raw messages, meaning that some non-identical messages are considered as repetition.

  For example, when the agent is running at `TRACE` level, the following message is written to the logs every 20 seconds when there is no other activity over the Websocket:

  ```
  TRACE 08:22:00 (20) Sending ping to check connection is still alive
  ```

  This message is produced by the logging statement:

  ```
  logger.trace("{} ({}) Sending ping to check connection is still alive", timeStamp, lastSent.getSeconds());
  ```

  Because the message text `{} ({}) Sending ping to check connection is still alive` is always the same, it is considered as repeat. Only the first 5 repeats are logged.
- Does not limit the lifespan of the cache. After the specified number of repetitions is reached, the repeated log messages never appear again, even if they are frequently hit.

Changing the log level in the AM admin UI

To change the log level in the AM admin UI

1. In the AM admin UI, go to **Realms** > realm name > **Application** > **Agents** > **Java**, and select your Java Agent.
2. On the **Global** tab, select **Agent Debug Level**, and select a level.

Alternatively, set Agent Debug Level as a custom property.

Notifications

AM sends the following notifications to Java Agent through WebSockets:

Configuration notifications

When the administrator makes a change to an agent configuration property, AM sends a notification to the agent. The agent flushes the configuration cache, and rereads the agent profile from AM. For more information about the cache, refer to Configuration cache.

Configuration notifications apply when you store the agent profile in AM's configuration data store. For information, refer to Enable Notifications of Agent Configuration Change.

Session notifications

When a client logs out, or a CTS-based session expires, AM sends a notification to the agent to remove that entry from the session cache. For information, refer to Enable Notification of Session Logout. For more information about the cache, refer to Session Cache.

Policy notifications

When an administrator changes a policy, AM sends a notification to the agent to flush the policy cache. For information, refer to Enable Notification of Policy Changes. For more information about the cache, refer to Policy Cache.

Notifications are enabled by default. To disable notifications, unsubscribe separately to each type of notification. If Autonomous mode is `true`, notifications and many other features are automatically disabled.

In configurations with load balancers and reverse proxies, make sure that the load balancers and reverse proxies support WebSockets.

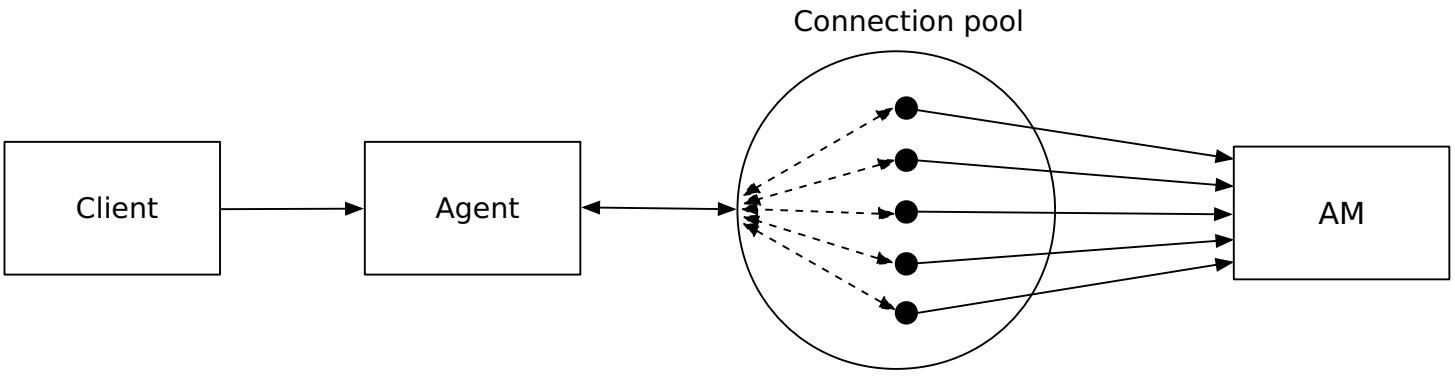
For more information about properties that configure notifications, refer to Notifications in the *Properties reference*.

Tuning connections

Use a connection pool between Java Agent and AM to reuse connections, and so reduce the overhead of creating new connections.

Connection pooling can improve performance in some circumstances. 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:

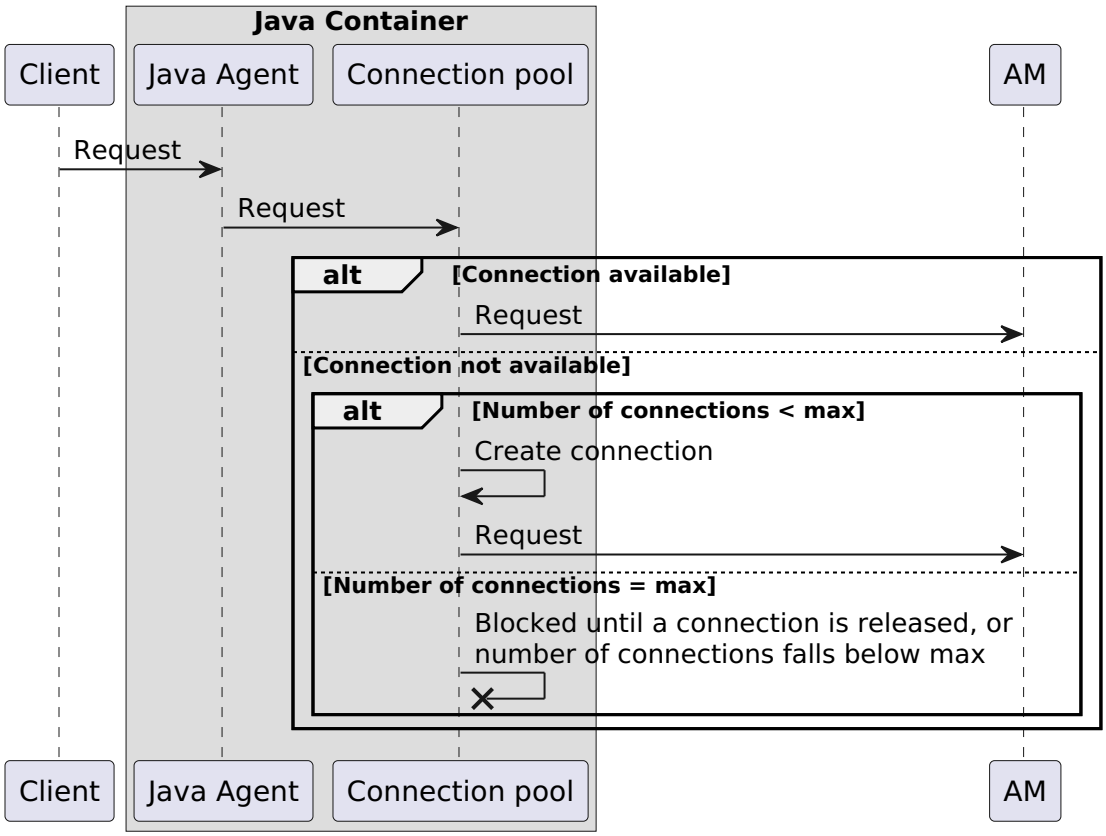


When a client makes a request, the agent intercepts the request, and uses the connection pool to connect to AM. If a connection is available, the agent uses that connection. The client is unaware of the connection reuse.

If a connection is not available, and the maximum number of connections is not reached, the agent creates and uses a new connection. When the maximum number of allowed connections is reached, the request waits until an existing connection is released, or a new connection can be created.

When the request is complete, the agent closes the connection to the pool but the physical connection is retained by the pool. The connection is then available to requests with the same connection parameters.

The following image shows the flow of information when a request is treated in a connection pool:



When Enable Connection Pooling is `true`, use the following properties to tune the connection pool:

Property	Description
Max HTTP Connection Count	By default, the connection pool can contain up to 1000 HTTP connections. Change the value of this property according to the AM load, as follows: • If AM is likely to be overloaded, reduce the maximum number of connections to reduce the load on AM. • If AM is not likely to be overloaded, use the default or a higher value so that connections are available when they are requested.
HTTP Connection Timeout	By default, connections wait for up to 10 seconds for a response before they are considered stale. Tune this property to allow enough time for systems to respond, and therefore prevent unnecessary failures, but to minimize the time to wait after a network failure.

Property	Description
HTTP Socket Timeout	By default, when a continuous data flow between the connection pool and AM is interrupted, the socket is considered stale. Tune this property to allow enough time for systems to respond, and therefore prevent unnecessary failures, but to minimize the time to wait after a socket failure.
Enable HTTP Connection Reuse	By default, after a request is complete the connection is returned to the connection pool. The physical connection is retained and can be reused by another request with the same connection parameters. Reuse connections to reduce the overhead of creating a new connection each time one is requested. If you are using the connection pool only as a throttling mechanism, to limit the total number of simultaneous connections, it is not necessary to reuse connections.
Enable HTTP Connection State	When Enable HTTP Connection Reuse is `true`, connection reuse is enabled by default for Apache HTTP Client.
Enable HTTP Retry	By default, requests are retried up to three times after failure. If requests are likely to pass on retry, enable this property to reduce the overhead of unnecessarily killing and remaking connections. If requests are likely to fail on retry, disable this property to reduce the retry time.

PingOne Advanced Identity Cloud guide

Ping 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 PingAM on-premise, or another on-premise access management solution. The guide provides an example of how to transition from on-premise access management to PingOne Advanced Identity Cloud without changing the architecture of the agent-based model.

The examples in this document are based on an available version of PingOne Advanced Identity Cloud. As PingOne Advanced Identity Cloud evolves, the examples in this document will be updated to reflect the changes.

Example installation for this guide

PingOne Advanced Identity Cloud is described in the PingOne Advanced Identity Cloud docs.

Find the value of the following properties:

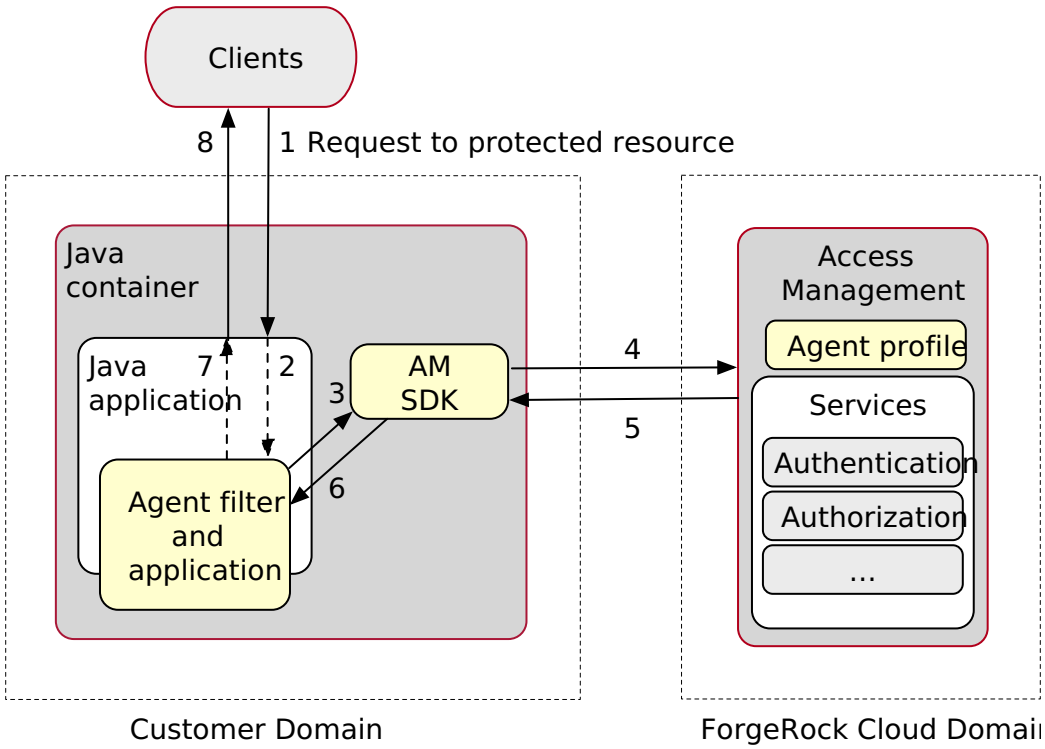
- The agent URL. This guide uses Java Agent installed on `http://agent.example.com:80`.
- The root URL of your PingOne Advanced Identity Cloud. This guide uses `https://tenant.forgeblocks.com`.
- The server URL of the PingAM component of the PingOne Advanced Identity Cloud. This guide uses `https://tenant.forgeblocks.com/am`.
- The realm where you work. This guide uses `alpha`.

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

About Java Agents and PingOne Advanced Identity Cloud

PingOne Advanced Identity Cloud simplifies the consumption of 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 Java Agent with the PingOne Advanced 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 Java Agent, and the Java Agent's interaction with PingOne Advanced Identity Cloud to enforce resource-based policies.



For information, refer to the PingOne Advanced Identity Cloud docs.

Prepare for installation

Learn more about installing Java Agent in Installation. Consider the following points for using the agent with PingOne Advanced Identity Cloud:

- Configure PingOne Advanced Identity Cloud and set up a policy before you install the agent. When you configure the agent in the Advanced 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 PingOne Advanced Identity Cloud servers, and between the agent and the client. Configure the environment **before** you install the agent.

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: `java-agent`
- Agent profile realm: `/alpha`
- Agent profile password: `/secure-directory/pwd.txt`

Add a demo user in PingOne Advanced Identity Cloud

Add a user so you can test the examples in this guide.

1. In the Advanced 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 PingOne Advanced Identity Cloud

1. In the Advanced 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 PingOne Advanced Identity Cloud

1. In the Advanced Identity Cloud admin UI, go to **Gateways & Agents** > **New Gateway/Agent**, and add a Java Agent with the following values:
   - **Agent ID** : `java-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 in the Access Management Native Console** to edit or create one.

4. (Optional) Use AM's secret service to manage the agent profile password. If AM finds a matching secret in a secret store, it uses that secret instead of the agent password configured in Step 1.

    1. In the settings panel of the agent profile page, click **Edit advanced settings in the Access Management Native Console**.
    2. In the AM admin UI, set a label for the agent password in **Secret Label Identifier**.

        AM uses the identifier to generate a secret label for the agent.

        The secret label has the format `am.application.agents.identifier.secret`, where identifier is the **Secret Label Identifier**.

        The **Secret Label Identifier** can only contain characters `a-z`, `A-Z`, `0-9`, and periods (`.`). It can't start or end with a period.
    3. Select **Secret Stores** and configure a secret store.
    4. Map the label to the secret. Learn more from AM's Map and rotate secrets.

    Note the following points for using AM's secret service:

    - Set a **Secret Label Identifier** that clearly identifies the agent.
    - If you update or delete the **Secret Label Identifier**, AM updates or deletes the corresponding mapping for the previous identifier provided no other agent shares the mapping.
    - When you rotate a secret, update the corresponding mapping.

Enforce policies decisions from PingOne Advanced Identity Cloud

This example sets up PingOne Advanced Identity Cloud as a policy decision point for requests processed by Java Agent. For more information about Java Agent, refer to the User guide.

1. Using the PingOne Advanced Identity Cloud docs, log in to PingOne Advanced Identity Cloud as an administrator.
2. Make sure that 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 PingOne Advanced Identity Cloud login page is displayed.
    2. Log in to PingOne Advanced Identity Cloud as user `demo`, password `Ch4ng3!t`, to access the web page protected by the Java Agent.

Security guide

Use this guide to reduce risk and mitigate threats to Java Agent 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.

Ping 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 Ping Identity Platform software common ways to access web resources and collections of resources.

Threats

The following sections describe some possible threats to Java Agent, 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 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, set Custom Response Header Map so that HTTP responses generated by the agent include the Cache-Control HTTP header. This header tells browsers and web proxies whether they can be cached. For example, clients can set the following value clear existing cache responses, force the cache to revalidate with the server, and prevent new responses from being cached:

```
org.forgerock.agents.response.header.map[Cache-Control]=max-age=0, no-store
```

In your decision to set this property, consider the impact on the performance of customer applications. Setting this property can reduce performance because browser pages are not cached.

For more information about caching, refer to HTTP caching.

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 identify Java Agent 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`.

- In custom login redirects, calls to the custom login URL include the parameter Login Reason Parameter Name to indicate why authentication is required, such as `NO_TOKEN` and `TOKEN_EXPIRED`. To reduce the risk of leaking useful information, use alternative strings for the authentication reason, by configuring Login Reason Value Map.

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 Java Agent 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. ForgeRock strongly recommends using token-based mitigations against CSRF, and relying on other measures only as a defense in depth, in accordance with OWASP guidance.

Configure the following properties to protect against cross-site scripting attacks:

- Enable Composite Advice Encoding
- XSS Redirect URI Map
- XSS Code Element List

POST data preservation

By default, POST data is stored in the in-memory cache. Consider the following points if you configure POST Data Preservation in Files or Cache to store POST data in the file system:

- Payloads from unauthenticated users are stored in the agent file system. If your threat evaluation does not accept this risk, store the data in the cache, or set POST Data Preservation in Files or Cache to `false`.
- Restrict access to the POST Data Preservation File Directory, to mitigate the risk of permissive access or leakage of personally identifiable information (PII).
- Limit the amount of stored POST data to mitigate the risk of DoS attacks, by configuring POST Data Preservation Storage Size or Max Entries in POST Data Preservation Storage.
- Remove expired POST data as soon as possible by configuring the POST Data Preservation Directory Sweep Interval.
- Identify threats in POST data before it is deleted, by making sure that Intrusion Detection Systems inspect the data within the time specified by POST Data Preservation Directory Sweep Interval.

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 password, refer to Preinstallation tasks. The encrypted password is stored in the AgentPassword.properties file.

> **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 using a password manager or by using the command line tool agentadmin --key.

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.

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 Java Agent software or the connected systems?

- How do you validate mitigation plans and remedial actions?
- How do client applications work when the Java Agent 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.

Denial of service

To prevent memory exhaustion DOS attacks, configure Maximum Decompression Size to limit the maximum size to which a compressed JWT can be decompressed. This property reduces the risk of a decompressed JWT consuming too much available memory.

Log overflow attacks

To prevent log overflow attacks, add a custom `agent-logback.xml` with a `DuplicateMessageFilter`. This filter detects duplicate messages, and after the specified number of repetitions, drops repeated messages. For more information, refer to Limiting repetitive log messages.

Operating systems

When you deploy Java Agent, 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 that you can expedite resolution of security issues.

To resolve security issues quickly, make sure that 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 Java Agent 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 Java Agent, such as unauthorized access to Java Agent files. Such violations are not recorded in Java Agent 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 Java Agent 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 Java Agent software securely include the following:

- A Java runtime environment, required to run Java Agent software.
- 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 that 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 Java Agent 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 Java Agent)

Protocol	Recommendations
HTTP	HTTP connections that are not protected by 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 TLS client authentication.

Recommendations For Outgoing Connections (From Java Agent to Another Service)

Client	Recommendations
Common Audit event handlers	Configure ForgeRock Common Audit event handlers to use HTTPS when connecting to external log services.
AM	Connect to AM over HTTPS, and use Web Socket Secure (WSS) for notifications. When AM listens on HTTPS, by default the agent uses WSS. Otherwise, by default the agent uses Web Sockets (WS).
Custom login pages	Connect to custom login pages over HTTPS.

Message-level security

Server protocols such as HTTP, LDAP, and JMX rely on TLS to protect connections. To enforce secure communication, refer to Secure communication between the agent and AM.

Communication between the agent and clients is managed by the web application container in which the agent runs. See the web application container documentation for information about how to secure those connections.

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 that only authorized people can access your servers and applications through the appropriate network, using the appropriate ports, and presenting strong-enough credentials.

Make sure that users connect to systems through the latest versions of TLS, and audit system access periodically.

Protect read-access to endpoints that monitor Common REST, Prometheus, CSV file-based metrics. For Common REST and Prometheus endpoints:

- Name exposed base endpoints to prevent them from being easily associated with an application.
- Set up strict not-enforced rules, to minimize unauthenticated access.

For information, refer to Manage endpoints for Common REST and Prometheus metrics.

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 Java Agent sessions

On startup, Java Agent uses the following properties to obtain a session from AM:

- Agent Profile Name
- Encrypted Agent 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 PingOne Advanced Identity Cloud and AM sessions

To minimize the time an attacker can attack an active session, set expiration timeouts for every PingOne Advanced 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 PingOne Advanced Identity Cloud:

- In the Advanced 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 AM:

- 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 Java Agent 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 pre-authentication cookies in Pre-Authentication Cookie Name, and JWT cookies in JWT Cookie Name.
- To transmit securely all cookies written by the agent, set Transmit Cookies Securely.
- To reduce the risk of cross-site request forgery (CSRF) attacks, set the SameSite attribute of cookies in Set-Cookie Internal Map or Set-Cookie Attribute Map.
- To ensure that cookies cannot be accessed through client-side scripts, and to mitigate any XSS attacks, set Enable HTTP Only Cookies 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.
- To protect the CDSSO session cookie from hijacking, configure AM as described in Enabling restricted tokens for CDSSO session cookies in AM's *Security guide*.

Keys and secrets

Java Agent 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.

In JVM, the default ephemeral Diffie-Hellman (DH) key size is 1024 bits. To support stronger ephemeral DH keys, and protect against weak keys, installations in Tomcat 8.5.37 and later versions use the Tomcat default DH key size of 2048-bit.

Increase the DH key size to protect against weak keys. For more information, refer to Customizing Size of Ephemeral Diffie-Hellman Keys

Create and 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.

Rotate the agent profile password

During installation, the agent requests the path to a file containing the agent profile password. The agent then uses the following properties to encrypt and store the password:

- `am.encryption.pwd` in the AgentKey.properties file
- `Encrypted Agent Password` in the AgentPassword.properties file

If the path is empty, the installation terminates with a configuration error.

The following steps describe how to rotate the agent profile password:

1. Change the profile password for your agent instance. For example, in the AM admin UI, change the password as follows:
    1. Select **Realms** > realm name > **Applications** > **Agents** > **Java**.
    2. Select your agent.
    3. In the **Global** tab, enter a new password in the **Password** field.

2. Generate an encryption key for the agent profile password, using the agentadmin --getEncryptKey command:

```
$ agentadmin --getEncryptKey
```

3. In AgentKey.properties, set the value of `am.encryption.pwd` to the new value.
4. Encrypt the agent profile password, using the agentadmin --encrypt command:

```
$ agentadmin --encrypt agent-instance password-file
```

The agent encrypts the password by using the value of `Encryption Key/Salt` from AgentKey.properties.

5. In AgentKey.properties, set the value of `Encrypted Agent Password` to the new value.
6. Restart the agent instance.

Create a cookie signing key

During installation, the agent requests the path to a file containing the cookie signing key, and then uses the key to configure the property `org.forgerock.agents.cookie.signing.value` in AgentKey.properties. If the path is empty, cookie signing is disabled.

The key must be at least 64 characters long. If it is shorter, the agent rejects it and leaves cookies unsigned. For security, use a key of at least 80 characters.

1. Generate an 80-character key, using the agentadmin --key command:

Unix

```
$ agentadmin --key 80
ZRY...xXO
```

Windows

```
C:> agentadmin --key 80
ZRY...xXO
```

Rotate cookie signing keys

1. Create a cookie signing key.
2. In AgentKey.properties, set the value of org.forgerock.agents.cookie.signing.value to the key value.

3. Restart the agent instance.

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 Ping Identity Platform. For information, refer to Auditing.

Java Agent supports propagation of the transaction ID across the Ping 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 Trust transaction headers in AM's *Security guide*.

Log files

Agent logs can contain informational, error, and warning events, to troubleshoot and debug transactions and events that take place within the agent instance.

Protect logs from unauthorised access, and make sure they contain a minimum of sensitive or personally identifiable information that could be used in attacks.

Use 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 aliases

A property alias specifies a path for a property. A property can have multiple aliases but each alias is unique to that property.

How the agent manages multiple aliases

When you assign multiple values to the same property through different aliases, the agent assigns the values as follows:

- For list properties, it appends each assignment to the list.
- For simple string properties, it overwrites the current value with each new value. The final value is the last value to be assigned.

The following example assigns different values to a string property with three aliases:

```
com.sun.identity.agents.app.username=one
com.sun.identity.agents.config.profilename=two
org.forgerock.agents.profile.name=three
```

The final value of the property is `three`, the last value to be assigned.

How AM manages multiple aliases

Each version of AM recognizes a different group of agent aliases. When you are using AM commands, such as `ssoadm` to configure an agent, consider the following points on using recognized and unrecognized aliases:

- When you use a recognized alias in an `ssoadm` command (for example, `com.sun.identity.agents.config.notenforced.ip.cache.size=2000`), the agent updates the value for the property represented by that alias.

  For the above example, **Max Entries in Not-Enforced IP Cache** is displayed as `2000` in the **Application** tab of the AM console.

- When you use an unrecognized alias in an `ssoadm` command (for example, `org.forgerock.agents.notenforced.ip.cache.size=4000`), the agent creates a custom property.

  For the above example, `org.forgerock.agents.notenforced.ip.cache.size=4000` is displayed in **Custom Properties**, in the **Advanced** tab of the AM console.

- When a property is set by both a standard property and a custom property, the custom property takes precedence. The value of the standard property is not updated, and both values are displayed in the configuration.

List properties

List properties can be configured with or without an index location. The following formats are allowed and equivalent:

```
property[0]=one
property[1]=two
property[2]=three
```

```
property=one
property=two
property=three
```

When the agent assigns values to a list property, it adds to the list in the order the assignments are made, ignoring any index specified, and appending to the end of the list. The following formats are equivalent:

```
property[]=one
property[]=two
property[]=three
```

```
property[10]=one
property[1]=two
property[42]=three
```

The agent uses the index location only in the following cases:

- When the index location is set to `@` and the value is comma-separated:

  The agent sets multiple properties according to the number of comma-separated values specified. In the following example, there are three comma-separated values:

  ```
  property[@]=one,two,three
  ```

  The agent sets three individual properties. The final assignment is as follows:

  ```
  property[]=one
  property[]=two
  property[]=three
  ```

- When the value for an index location is empty:

  The agent deletes that location in the list. In the following example, the last value for index location `[1]` is empty:

```
property[0]=one
property[1]=two
property[2]=three
property[1]=
```

The agent deletes index location `[1]` from the list and then moves index location `[2]` to `[1]`. The final assignment is as follows:

```
property[0]=one
property[1]=three
```

- When the index location is empty and the value is empty:

  The agent deletes all values from the list; the list exists, but is empty. In the following example, the second value for index location `[]` is empty:

  ```
  property[]=one
  property[]=
  property[]=two
  property[]=three
  ```

  The agent does the following:

  - Adds the text "one" to the list
  - Deletes all values from the list
  - Adds the text "two" into index location `[0]`
  - Adds the text "three" into index location `[1]`

  The final assignment is as follows:

  ```
  property[0]=two
  property[1]=three
  ```

List of bootstrap properties

Property	Description	Function
org.forgerock.agents.filter.mode.map	Agent Filter Mode Map	Agent
org.forgerock.agents.profile.name	Agent Profile Name	Profile, Required
org.forgerock.agents.agent.profile.realm	Agent Profile Realm	Profile, Required

Property	Description	Function
org.forgerock.agents.am.hostname	AM Authentication Service Host Name	Authentication service, Required
org.forgerock.agents.am.path	AM Authentication Service Path	Authentication service, Required
org.forgerock.agents.am.port	AM Authentication Service Port	Authentication service, Required
org.forgerock.agents.am.protocol	AM Authentication Service Protocol	Authentication service, Required
org.forgerock.agents.fallback.mode.enabled	Autonomous mode	Agent, Required
org.forgerock.agents.csv.monitoring.directory	CSV Monitoring Directory	Monitoring
org.forgerock.agents.lock.config.enabled	Enable Configuration Lock	Profile
org.forgerock.agents.http.client.reuse.connections.enabled	Enable HTTP Connection Reuse	Connection pooling
org.forgerock.agents.http.client.connection.state.enabled	Enable HTTP Connection State	Connection pooling
org.forgerock.agents.http.client.retry.requests.enabled	Enable HTTP Retry	Connection pooling
org.forgerock.agents.policy.change.notifications.enabled	Enable Notification of Policy Changes	Notifications
org.forgerock.agents.config.change.notifications.enabled	Enable Notifications of Agent Configuration Change	Notifications
org.forgerock.agents.prometheus.monitoring.enabled	Enable Prometheus Monitoring	Global
org.forgerock.agents.encryptor.classname	Encryption Class	Encryption, Required
org.forgerock.agents.sso.exchange.cache.ttl.minutes	Exchanged SSO Token Cache TTL	Profile
org.forgerock.agents.sso.expired.session.cache.ttl.minutes	Expired Session Cache Timeout	Session
org.forgerock.agents.http.client.connection.timeout.seconds	HTTP Connection Timeout	Connection pooling

Property	Description	Function
org.forgerock.agents.http.client.socket.timeout.seconds	HTTP Socket Timeout	Connection pooling
org.forgerock.agents.not.enforced.rule.pattern.matcher.classname	Java Class for Matching Not Enforced Rules	Not-enforced
org.forgerock.agents.jwt.cache.ttl.minutes	JWT Cache TTL	Profile
org.forgerock.agents.jwt.cookie.name	JWT Cookie Name	Profile
org.forgerock.agents.local.audit.file.path	Local Audit Log Filename	Audit
org.forgerock.agents.config.location	Location of Agent Configuration Repository	Profile, Required
org.forgerock.agents.expired.session.cache.size	Max Entries in Expired Session Cache	Session
org.forgerock.agents.jwt.cache.size	Max Entries in JWT Cache	Profile
org.forgerock.agents.policy.cache.per.session.size	Max Entries in Policy Cache per Session	Policy enforcement
org.forgerock.agents.pdp.cache.size	Max Entries in POST Data Preservation Storage	POST data preservation
org.forgerock.agents.sso.exchange.cache.size	Max Entries in SSO Exchange Cache	Profile
org.forgerock.agents.http.client.max.connections	Max HTTP Connection Count	Connection pooling
org.forgerock.agents.policy.cache.session.size	Max Sessions in Policy Cache	Policy enforcement
org.forgerock.agents.auto.not.enforce.favicon	Not-Enforced Favicon	Not-enforced
org.forgerock.agents.policy.cache.ttl.minutes	Policy Cache TTL	Policy enforcement
org.forgerock.agents.pdp.cache.ttl.minutes	POST Data Preservation Cache TTL	POST data preservation
com.sun.identity.agents.config.postdata.preserve.cache.entry.ttl	POST Data Preservation Cache TTL in Milliseconds (deprecated)	POST data preservation
org.forgerock.agents.pdp.directory.sweep.seconds	POST Data Preservation Directory Sweep Interval	POST data preservation

Property	Description	Function
org.forgerock.agents.pdp.cache.total.size.mb	POST Data Preservation Storage Size	POST data preservation
org.forgerock.agents.public.am.url	Public AM URL	Miscellaneous, Required
org.forgerock.agents.service.resolver.class.name	Service Resolver Class Name	Miscellaneous
org.forgerock.agents.session.cache.ttl.minutes	Session Cache TTL	Session
org.forgerock.agents.strategy.when.am.unavailable	Strategy when AM unavailable	Agent

List of all properties

Property	Description (UI name)	Function
org.forgerock.agents.access.denied.uri.map	Access Denied URI Map	Access denied
org.forgerock.agents.debug.level	Agent Debug Level	Logs
org.forgerock.agents.filter.mode.map	Agent Filter Mode Map	Agent
org.forgerock.agents.profile.name	Agent Profile Name	Profile, Required
org.forgerock.agents.agent.profile.realm	Agent Profile Realm	Profile, Required
org.forgerock.agents.agent.hostname	Alternative Agent Host Name	Agent
org.forgerock.agents.agent.port	Alternative Agent Port Number	Agent
org.forgerock.agents.agent.protocol	Alternative Agent Protocol	Agent
org.forgerock.agents.config.logout.session.invalidate.enabled	Always invalidate sessions	Logout
org.forgerock.agents.am.hostname	AM Authentication Service Host Name	Authentication service, Required

Property	Description (UI name)	Function
org.forgerock.agents.am.path	AM Authentication Service Path	Authentication service, Required
org.forgerock.agents.am.port	AM Authentication Service Port	Authentication service, Required
org.forgerock.agents.am.protocol	AM Authentication Service Protocol	Authentication service, Required
com.sun.identity.agents.config.login.url	AM Login URL List	Custom login redirect, Default Login Redirect, Login redirect, Login Redirect (Default)
org.forgerock.agents.audit.what	Audit Access Types	Audit
org.forgerock.agents.local.audit.log.retention.count	Audit Logfile Retention Count	Audit
org.forgerock.agents.audit.where	Audit Log Location	Audit
org.forgerock.agents.authn.exchange.cookie.name	Authentication Exchange Cookie Name	Login
org.forgerock.agents.authn.exchange.uri	Authentication Exchange URI	Login
org.forgerock.agents.authn.fail.reason.parameter.name	Authentication Fail Reason Parameter Name	Authentication failure
org.forgerock.agents.authn.fail.reason.remapper	Authentication Fail Reason Parameter Value Map	Authentication failure
org.forgerock.agents.authn.fail.url	Authentication Fail URL	Authentication failure
org.forgerock.agents.authn.redirect.uri	Authentication Redirect URI	Cross-domain single sign-on, Required
org.forgerock.agents.fallback.mode.enabled	Autonomous mode	Agent, Required
org.forgerock.agents.bad.advice.loop.termination.counter	Bad advice loop termination counter	Bad configuration detection

Property	Description (UI name)	Function
org.forgerock.agents.bad.advice.loop.termination.http.code	Bad advice loop termination HTTP status	Bad configuration detection
org.forgerock.agents.bad.advice.loop.termination.url	Bad advice loop termination URL	Bad configuration detection
org.forgerock.agents.http.header.containing.remote.hostname	Client Hostname Header	Client identification, Continuous security
org.forgerock.agents.http.header.containing.ip.address	Client IP Address Header	Client identification, Continuous security
org.forgerock.agents.acceptable.ip.address.map	Client IP Validation Address Map	Client identification
org.forgerock.agents.original.ip.check.mode.map	Client IP Validation Mode	Client identification
org.forgerock.agents.conditional.logout.url.list	Conditional Logout URL List	Logout
org.forgerock.agents.config.reload.seconds	Configuration Reload Interval	Profile
org.forgerock.agents.container.encoding	Container Character Encoding	Container, Not-enforced
org.forgerock.agents.container.param.encoding	Container Parameter Encoding	Container, Not-enforced
org.forgerock.agents.continuous.security.cookies.map	Continuous Security Cookie Map	Continuous security
org.forgerock.agents.continuous.security.headers.map	Continuous Security Header Map	Continuous security
org.forgerock.agents.accept.ipdp.cookie	Convert SSO Tokens Into OIDC JWTs	SSO cookie handling
org.forgerock.agents.cookie.reset.enabled	Cookie Reset	Cookie reset
org.forgerock.agents.attribute.cookie.separator	Cookie Separator Character	Attributes
org.forgerock.agents.csv.monitoring.directory	CSV Monitoring Directory	Monitoring

Property	Description (UI name)	Function
org.forgerock.agents.response.header.map	Custom Response Header Map	Miscellaneous
org.forgerock.agents.fqdn.default	Default FQDN	Fully qualified domain name
org.forgerock.agents.attribute.cookie.encode.enabled	Enable Attribute Encoding	Attributes
org.forgerock.agents.advice.b64.url.encode.enabled	Enable Composite Advice Encoding	Policy enforcement
org.forgerock.agents.lock.config.enabled	Enable Configuration Lock	Profile
org.forgerock.agents.use.connection.pooling.enabled	Enable Connection Pooling	Connection pooling
org.forgerock.agents.legacy.login.enabled	Enable Custom Login Mode	Custom login redirect, Default Login Redirect, Login redirect, Login Redirect (Default)
com.iplanet.am.cookie.encode	Enable Encoded Cookies	Cookie
org.forgerock.agents.fqdn.check.enabled	Enable FQDN Checking	Fully qualified domain name
org.forgerock.agents.302.redirects.enabled	Enable HTTP 302 Redirects	Global
org.forgerock.agents.http.client.reuse.connections.enabled	Enable HTTP Connection Reuse	Connection pooling
org.forgerock.agents.http.client.connection.state.enabled	Enable HTTP Connection State	Connection pooling
com.sun.identity.cookie.httponly	Enable HTTP Only Cookies	Cookie
org.forgerock.agents.http.client.retry.requests.enabled	Enable HTTP Retry	Connection pooling
org.forgerock.agents.ignore.path.info.enabled	Enable Ignore Path Info	Miscellaneous

Property	Description (UI name)	Function
org.forgerock.agents.load.balancer.cookies.enabled	Enable Load Balancer Cookies	Cookie
org.forgerock.agents.local.audit.log.rotation.enabled	Enable Local Audit Log Rotation	Audit
org.forgerock.agents.logout.introspection.enabled	Enable Logout Introspection	Logout
org.forgerock.agents.notenforced.ip.cache.enabled	Enable Not-Enforced IP Cache	Not-enforced
org.forgerock.agents.notenforced.uri.cache.enabled	Enable Not-Enforced URIs Cache	Not-enforced
org.forgerock.agents.policy.change.notifications.enabled	Enable Notification of Policy Changes	Notifications
com.iplanet.am.session.client.polling.enabled	Enable Notification of Session Logout (deprecated)	Notifications
org.forgerock.agents.session.change.notifications.enabled	Enable Notification of Session Logout	Notifications
org.forgerock.agents.config.change.notifications.enabled	Enable Notifications of Agent Configuration Change	Notifications
org.forgerock.agents.user.realm.overrides.policy.evaluation.realm.enabled	Enable Policy Evaluation in User Authentication Realm	Policy enforcement
org.forgerock.agents.post.data.preservation.enabled	Enable POST Data Preservation	POST data preservation
org.forgerock.agents.prometheus.monitoring.enabled	Enable Prometheus Monitoring	Global
org.forgerock.agents.authn.success.redirect.session.url.enabled	Enable Redirect to AM Success URL	Login
org.forgerock.agents.accept.sso.tokens.enabled	Enable SSO Token Acceptance	Custom login redirect, Login redirect, SSO cookie handling

Property	Description (UI name)	Function
org.forgerock.agents.userid.mapping.mode.use.dn.enabled	Enable User Principal Flag	User mapping
org.forgerock.agents.encrypted.password	Encrypted Agent Password	Miscellaneous, Required
org.forgerock.agents.encryptor.classname	Encryption Class	Encryption, Required
org.forgerock.agents.encryption.key	Encryption Key/Salt	Authentication service, Encryption
org.forgerock.agents.sso.exchange.cache.ttl.minutes	Exchanged SSO Token Cache TTL	Profile
org.forgerock.agents.samesite.excluded.user.agents.list	Exclude Agents From Samesite Cookie Attributes	SameSite
org.forgerock.agents.sso.expired.session.cache.ttl.minutes	Expired Session Cache Timeout	Session
org.forgerock.agents.config.monitoring.to.csv	Export Monitoring Metrics to CSV	Monitoring
org.forgerock.agents.attribute.date.format	Fetch Attribute Date Format	Attributes
org.forgerock.agents.fqdn.map	FQDN Map	Fully qualified domain name
org.forgerock.agents.authn.fragment.relay.uri	Fragment Relay URI	Fragment
org.forgerock.agents.continuous.security.get.list	GET Parameter List for URL Policy Env	Policy enforcement
com.sun.identity.agents.config.redirect.param	Goto Parameter Name	Global
org.forgerock.agents.default.goto.url	Goto URL	Authentication failure
org.forgerock.agents.302.redirect.http.content.type	HTTP 302 Redirect Content Type	Global
org.forgerock.agents.302.redirect.http.data	HTTP 302 Redirect Data	Global

Property	Description (UI name)	Function
org.forgerock.agents.config.json.url.invert	HTTP 302 Redirect Invert Not-Enforced List	Global
org.forgerock.agents.config.json.url	HTTP 302 Redirect Not-Enforced List	Global
org.forgerock.agents.302.redirect.http.status.code	HTTP 302 Redirect Replacement HTTP Status Code	Global
org.forgerock.agents.http.client.connection.timeout.seconds	HTTP Connection Timeout	Connection pooling
org.forgerock.agents.http.session.binding.enabled	HTTP Session Binding	Miscellaneous
org.forgerock.agents.http.client.socket.timeout.seconds	HTTP Socket Timeout	Connection pooling
org.forgerock.agents.idle.time.window.minutes	Idle Time Refresh Window	Miscellaneous
org.forgerock.agents.notenforced.ip.invert.enabled	Invert Not-Enforced IPs	Not-enforced
org.forgerock.agents.notenforced.uri.invert.enabled	Invert Not-Enforced URIs	Not-enforced
org.forgerock.agents.not.enforced.rule.pattern.matcher.classname	Java Class for Matching Not Enforced Rules	Not-enforced
org.forgerock.agents.continuous.security.http.session.list	JSession Parameter List for URL Policy Env	Policy enforcement
org.forgerock.agents.jwt.cache.ttl.minutes	JWT Cache TTL	Profile
org.forgerock.agents.jwt.cookie.domain.list	JWT Cookie Domain List	Profile
org.forgerock.agents.jwt.cookie.name	JWT Cookie Name	Profile
org.forgerock.agents.legacy.login.url.list	Legacy Login URL List	Custom login redirect, Default Login Redirect, Login redirect, Login Redirect (Default)

Property	Description (UI name)	Function
org.forgerock.agents.load.balancer.cookie.name	Load Balancer Cookie Name	Cookie
org.forgerock.agents.local.audit.file.path	Local Audit Log Filename	Audit
org.forgerock.agents.local.audit.log.rotation.bytes	Local Audit Log Rotation Size	Audit
org.forgerock.agents.locale.country	Locale Country	Locale
org.forgerock.agents.locale.language	Locale Language	Locale
org.forgerock.agents.config.location	Location of Agent Configuration Repository	Profile, Required
org.forgerock.agents.login.attempt.limit.count	Login Attempt Limit (deprecated)	Deprecated
org.forgerock.agents.login.counter.cookie.name	Login Attempt Limit Cookie Name (deprecated)	Deprecated
org.forgerock.agents.login.reason.parameter.name	Login Reason Parameter Name	Custom login redirect, Login redirect
org.forgerock.agents.login.reason.remapper	Login Reason Value Map	Login
org.forgerock.agents.logout.goto.map	Logout Entry URI Map	Logout
org.forgerock.agents.logout.request.param.map	Logout Request Parameter Map	Logout
org.forgerock.agents.logout.endpoint.map	Logout URI Map	Logout
org.forgerock.agents.authn.cookie.max.age.seconds	Max Age of Pre-Authentication Cookie	Cookie, Pre-authentication
org.forgerock.agents.expired.session.cache.size	Max Entries in Expired Session Cache	Session

Property	Description (UI name)	Function
org.forgerock.agents.jwt.cache.size	Max Entries in JWT Cache	Profile
org.forgerock.agents.notenforced.ip.cache.size	Max Entries in Not-Enforced IP Cache	Not-enforced
org.forgerock.agents.notenforced.uri.cache.size	Max Entries in Not-Enforced URI Cache	Not-enforced
org.forgerock.agents.policy.cache.per.session.size	Max Entries in Policy Cache per Session	Policy enforcement
org.forgerock.agents.pdp.cache.size	Max Entries in POST Data Preservation Storage	POST data preservation
org.forgerock.agents.sso.exchange.cache.size	Max Entries in SSO Exchange Cache	Profile
org.forgerock.agents.http.client.max.connections	Max HTTP Connection Count	Connection pooling
org.forgerock.agents.max.decompression.size.bytes	Maximum Decompression Size	Cookie
org.forgerock.agents.policy.cache.session.size	Max Sessions in Policy Cache	Policy enforcement
org.forgerock.agents.pdp.noentry.url.map	Missing POST Data Preservation Entry URI Map	POST data preservation
org.forgerock.agents.notenforced.ip.list	Not-Enforced Client IP List	Not-enforced
org.forgerock.agents.notenforced.compound.separator	Not-Enforced Compound Rule Separator	Not-enforced
org.forgerock.agents.auto.not.enforce.favicon	Not-Enforced Favicon	Not-enforced
org.forgerock.agents.notenforced.uri.list	Not-Enforced URIs	Not-enforced

Property	Description (UI name)	Function
org.forgerock.agents.oauth.login.url.list	OAuth Login URL List	Custom login redirect, Default Login Redirect, Login redirect, Login Redirect (Default)
org.forgerock.agents.policy.cache.ttl.minutes	Policy Cache TTL	Policy enforcement
org.forgerock.agents.policy.evaluation.realm.map	Policy Evaluation Realm Map	Policy enforcement
org.forgerock.agents.policy.set.map	Policy Set Map	Policy enforcement
org.forgerock.agents.pdp.cache.ttl.minutes	POST Data Preservation Cache TTL	POST data preservation
com.sun.identity.agents.config.postdata.preserve.cache.entry.ttl	POST Data Preservation Cache TTL in Milliseconds (deprecated)	POST data preservation
org.forgerock.agents.pdp.cookie.name	Post Data Preservation Cookie Name	Cookie, POST data preservation
org.forgerock.agents.pdp.directory.sweep.seconds	POST Data Preservation Directory Sweep Interval	POST data preservation
org.forgerock.agents.pdp.directory	POST Data Preservation File Directory	POST data preservation
org.forgerock.agents.pdp.use.filesystem.enabled	POST Data Preservation in Files or Cache	POST data preservation
org.forgerock.agents.pdp.parameter.limit.bytes	POST Data Preservation Max HTML Form Size	POST data preservation
org.forgerock.agents.pdp.sticky.session.value	POST Data Preservation Sticky Session Key Value	POST data preservation

Property	Description (UI name)	Function
org.forgerock.agents.pdp.sticky.session.mode	POST Data Preservation Sticky Session Mode	POST data preservation
org.forgerock.agents.pdp.cache.total.size.mb	POST Data Preservation Storage Size	POST data preservation
org.forgerock.agents.continuous.security.post.list	POST Parameter List for URL Policy Env	Policy enforcement
org.forgerock.agents.authn.cookie.name	Pre-Authentication Cookie Name	Cookie, Pre-authentication
org.forgerock.agents.cookie.signing.value	Pre-Authn and Post Data Preservation Cookie Signing Value	Cookie, POST data preservation, Pre-authentication
org.forgerock.agents.profile.attribute.fetch.mode	Profile Attribute Fetch Mode	Attributes, Cookie reset, Profile
org.forgerock.agents.profile.attribute.map	Profile Attribute Map	Profile
org.forgerock.agents.public.am.url	Public AM URL	Miscellaneous, Required
org.forgerock.agents.wanted.http.url.param.list	Query Parameter List for Policy Evaluation	Query parameter
org.forgerock.agents.am.unavailability.recheck.window.in.seconds	Recheck availability of AM	Agent
org.forgerock.agents.redirect.cookie.name	Redirect Attempt Cookie Name	Miscellaneous
org.forgerock.agents.redirect.attempt.limit	Redirect Attempt Limit	Login
org.forgerock.agents.wanted.http.url.params.regexp.list	Regex Query Parameters List for Policy Evaluation	Query parameter
org.forgerock.agents.unwanted.http.url.params.regex.list	Regex Remove Query Parameters List for Policy Evaluation	Query parameter

Property	Description (UI name)	Function
org.forgerock.agents.unwanted.http.url.param.list	Remove Query Parameters List for Policy Evaluation	Query parameter
org.forgerock.agents.cookie.reset.domain.map	Reset Cookie Domain Map	Cookie reset
org.forgerock.agents.cookie.reset.name.list	Reset Cookie List	Cookie reset
org.forgerock.agents.cookie.reset.path.map	Reset Cookie Path Map	Cookie reset
org.forgerock.agents.response.attribute.fetch.mode	Response Attribute Fetch Mode	Attributes, Response
org.forgerock.agents.response.attribute.map	Response Attribute Map	Response
org.forgerock.agents.restrict.to.realm.map	Restrict to Realm Map	Policy enforcement
org.forgerock.agents.retain.previous.override.behaviour.enabled	Retain previous override behaviour	Configure behaviour
org.forgerock.agents.service.resolver.class.name	Service Resolver Class Name	Miscellaneous
org.forgerock.agents.session.attribute.fetch.mode	Session Attribute Fetch Mode	Attributes, Cookie reset, Session
org.forgerock.agents.session.attribute.map	Session Attribute Map	Session
org.forgerock.agents.session.cache.ttl.minutes	Session Cache TTL	Session
org.forgerock.agents.set.cookie.attribute.map	Set-Cookie Attribute Map	SameSite
org.forgerock.agents.set.cookie.internal.map	Set-Cookie Internal Map	SameSite
org.forgerock.agents.ipdp.cookie.domain.list	SSO Cookie Domain List	SSO cookie handling
org.forgerock.agents.strategy.when.am.unavailable	Strategy when AM unavailable	Agent
org.forgerock.agents.secure.cookies.enabled	Transmit Cookies Securely	Cross-domain single sign-on

Property	Description (UI name)	Function
org.forgerock.agents.user.mapping.mode.attribute.name	User Attribute Name	User mapping
org.forgerock.agents.user.mapping.mode	User Mapping Mode	User mapping
org.forgerock.agents.userid.mapping.mode.use.session.property.name	User Session Name	User mapping
org.forgerock.agents.balance.websocket.interval.minutes	WebSocket Connection Interval	Profile
org.forgerock.agents.declare.websocket.dead.after.milliseconds	Websocket Expired Timeout	Timeout
org.forgerock.agents.ping.websocket.after.inactive.milliseconds	Websocket Idle Timeout	Timeout
org.forgerock.agents.xss.code.element.list	XSS Code Element List	Cross-site scripting
org.forgerock.agents.xss.redirect.uri.map	XSS Redirect URI Map	Cross-site scripting

Access denied

Access Denied URI Map

The URIs of custom pages to return when access is denied. The key is the web application name. The value is the custom URI.

To set a global custom access denied URI for web applications without other custom access denied URIs defined, leave the key empty and set the value to the global custom access denied URI, `/s6ample/accessdenied.html`.

To set a custom access denied URI for a specific web application, set the key to the name of the web application, and the value to the web application access denied URI, such as `/myApp/accessdenied.html`.

Specify a full URL if required, including the host name. For example: `https://help.example.com/errors/accessdenied.html`.

Property name	`org.forgerock.agents.access.denied.uri.map`

Aliases	`com.sun.identity.agents.config.access.denied.uri`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.access.denied.uri.map`
	Introduced in Java Agent 5.6

Function	Access denied

Type	Map • Keys: web application • Values: URI of page saying 'access denied'
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Application` Title: `Access Denied URI Map` Legacy title: `Resource Access Denied URI`

Agent

Alternative Agent Protocol

In environments when agents are behind a load balancer or reverse proxy which does a SSL offloading, the request URL is changed to match the URL that the agent receives.

The agent then uses the new URL as the redirection value in the pre-authentication cookie, created during the first unauthenticated request to the agent.

Use the following properties to override the agent redirection value with the URL of the original client request: Alternative Agent Host Name, and Alternative Agent Port Number.

Property name	`org.forgerock.agents.agent.protocol`

Aliases	`com.sun.identity.agents.config.agent.protocol`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.agent.protocol`
	Introduced in Java Agent 5.6

Function	Agent
Type	String
Bootstrap property	No
Required property	No
Restart required	No

Local configuration file	`AgentConfig.properties`
AM console	Tab: `Advanced` Title: `Alternative Agent Protocol`

Autonomous mode

When `true` the agent operates independently of AM, without needing to contact an AM instance. Agents allow access to resources as defined in not-enforced lists; otherwise, they deny access.

Property name	`org.forgerock.agents.fallback.mode.enabled`

Aliases	`com.forgerock.agents.config.fallback.mode`
	Introduced in Java Agent 5.9.0
	`org.forgerock.agents.fallback.mode.enabled`
	Introduced in Java Agent 5.9.0
	`org.forgerock.agents.autonomous.mode.enabled`
	Introduced in Java Agent 5.9.0

Function	Agent, Required
Type	Boolean: `true` returns true; all other strings return `false`.
Default	`false`
Bootstrap property	Yes
Required property	Yes - If this property is missing, the agent fails to start
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentBootstrap.properties`

Recheck availability of AM

The duration after which the agent rechecks AM availability, when Autonomous mode is `false`, and AM becomes unavailable at runtime.

Consider these points when you configure this property:

- If the duration is too short, the agent checks AM availability too often, and agent performance can be reduced.
- If the duration is zero, the agent checks AM availability for every call. Requests that match not-enforced rules can take longer.

Property name	`org.forgerock.agents.am.unavailability.recheck.window.in.seconds`

Aliases	`org.forgerock.agents.am.unavailability.recheck.window.in.seconds` Introduced in Java Agent 5.9.0 Recognized from AM 7.2
Function	Agent
Type	Integer
Default	`5`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Profile (from AM 7.2)` Title: `Recheck availability of AM`

Alternative Agent Port Number

In environments when agents are behind a load balancer or reverse proxy which does a SSL offloading, the request URL is changed to match the URL that the agent receives.

The agent then uses the new URL as the redirection value in the pre-authentication cookie, created during the first unauthenticated request to the agent.

Use the following properties to override the agent redirection value with the URL of the original client request: Alternative Agent Host Name, and Alternative Agent Protocol.

Property name	`org.forgerock.agents.agent.port`

Aliases	`com.sun.identity.agents.config.agent.port`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.agent.port`
	Introduced in Java Agent 5.6

Function	Agent
Type	Integer
Default	`-2147483648`
Bootstrap property	No

Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Advanced` Title: `Alternative Agent Port Number`

Alternative Agent Host Name

In environments when agents are behind a load balancer or reverse proxy which does a SSL offloading, the request URL is changed to match the URL that the agent receives.

The agent then uses the new URL as the redirection value in the pre-authentication cookie, created during the first unauthenticated request to the agent.

Use the following properties to override the agent redirection value with the URL of the original client request: Alternative Agent Port Number and Alternative Agent Protocol.

Property name	`org.forgerock.agents.agent.hostname`

Aliases	`com.sun.identity.agents.config.agent.host`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.agent.hostname`
	Introduced in Java Agent 5.6

Function	Agent
Type	String
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Advanced` Title: `Alternative Agent Host Name`

Agent Filter Mode Map

A map of web application name to agent filter mode:

- Key: Web application name.
- Value: Agent filter mode.

The following example configures one filter mode for the `BankApp` web application. All other web applications use the default filter mode, `URL_POLICY`: `org.forgerock.agents.filter.mode.map[BankApp]=SSO_ONLY`

The following example configures the `NONE` filter mode for all applications in the web container: `org.forgerock.agents.filter.mode.map[ALL]=NONE`

The mode `J2EE_POLICY` does not apply to Java Agents 5.10. However, for backward-compatibility, it is displayed in the AM agent profile page.

Property name	`org.forgerock.agents.filter.mode.map`

Aliases	`com.sun.identity.agents.config.filter.mode`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.filter.mode.map`
	Introduced in Java Agent 5.6.2.1

Function	Agent
Supported settings	***NONE*** The agent performs no authentication check, and grants access to any resource. When the value is `NONE` and logging is enabled, for auditing purposes, the agent filter logs all incoming requests. ***SSO_ONLY*** Any user having either a valid SSO token or JWT can access any resource. ***URL_POLICY*** The normal operating mode of the agent, in which resource access is granted by AM policy evaluation.
Default	`URL_POLICY`
Bootstrap property	Yes
Required property	No
Restart required	No
Local configuration file	`AgentBootstrap.properties`

AM console	Tab: `Global` Title: `Agent Filter Mode Map` Legacy title: `Agent Filter Mode`

Strategy when AM unavailable

When Autonomous mode is `false`, this property defines the strategy to use when AM becomes unavailable at runtime (for example, due to network errors).

Default: `EVAL_NER_USE_CACHE_UNTIL_EXPIRED_ELSE_503`

Property name	`org.forgerock.agents.strategy.when.am.unavailable`
Aliases	`org.forgerock.agents.strategy.when.am.unavailable` Introduced in Java Agent 5.9.0
Function	Agent

Supported settings	***IMMEDIATE_403*** When AM is unavailable, immediately return HTTP 403 for every request ***IMMEDIATE_503*** When AM is unavailable, immediately return HTTP 503 for every request ***EVAL_NER_ELSE_403*** When AM is unavailable, match incoming requests against not-enforced rules. Grant access to matched resources. Return HTTP 403 for all other requests. ***EVAL_NER_ELSE_503*** When AM is unavailable, match incoming requests against not-enforced rules. Grant access to matched resources. Return HTTP 503 for all other requests. ***EVAL_NER_USE_CACHE_UNTIL_EXPIRED_ELSE_403*** When AM is unavailable, match incoming requests against not-enforced rules. Resolve unmatched requests against the cache. Return HTTP 403 for requests that don't match the cache result. Cached entries expire naturally. After the interval defined in "Policy Cache TTL" (org.forgerock.agents.policy.cache.ttl.minutes), this becomes exactly like EVAL_NER_ELSE_403. ***EVAL_NER_USE_CACHE_UNTIL_EXPIRED_ELSE_503*** When AM is unavailable, match incoming requests against not-enforced rules. Resolve unmatched requests against the cache. Return HTTP 503 for requests that don't match the cache result. Cached entries expire naturally. After the interval defined in "Policy Cache TTL" (org.forgerock.agents.policy.cache.ttl.minutes), this becomes exactly like EVAL_NER_ELSE_503. ***EVAL_NER_CACHE_INDEFINITELY_ELSE_403*** As soon as AM becomes unavailable, freeze values in the agent caches and preserve them indefinitely. Match incoming requests against not-enforced rules. Resolve unmatched requests against the frozen cache. Return HTTP 403 for requests that don't match the cache result. ***EVAL_NER_CACHE_INDEFINITELY_ELSE_503*** As soon as AM becomes unavailable, freeze values in the agent caches and preserve them indefinitely. Match incoming requests against not-enforced rules. Resolve unmatched requests against the frozen cache. Return HTTP 503 for requests that don't match the cache result.
Default	`EVAL_NER_USE_CACHE_UNTIL_EXPIRED_ELSE_503`
Bootstrap property	Yes
Required property	No
Restart required	No
Local configuration file	`AgentBootstrap.properties`

Attributes

Session Attribute Fetch Mode

Map the name of an AM session attribute specified in Session Attribute Map as follows:

- NONE: Do not map
- HTTP_HEADER: Map the to the name of an HTTP session header
- HTTP_COOKIE: Map to the name of an HTTP cookie
- REQUEST_ATTRIBUTE: Map to the name of an HTTP session attribute

When the value is `HTTP_COOKIE`, Cookie Reset is automatically set to `true`. Before redirecting the client for login, and when the client logs out, the agent resets profile and session attributes cookies, and cookies in the Reset Cookie List.

Property name	`org.forgerock.agents.session.attribute.fetch.mode`

Aliases	`org.forgerock.agents.session.attribute.fetch.mode`
	Introduced in Java Agent 5.6
	`com.sun.identity.agents.config.session.attribute.fetch.mode`
	Introduced in Java Agent 5.0
	Recognized from AM 6

Function	Attributes, Cookie reset, Session
Supported settings	***NONE*** No AM attributes are mapped. ***HTTP_HEADER*** The named AM attributes are mapped to headers in the outgoing HTTP response. ***REQUEST_ATTRIBUTE*** The named AM attributes are mapped to request attributes in the outgoing HTTP response. ***HTTP_COOKIE*** The named AM attributes are mapped to cookies in the outgoing HTTP response.
Default	`NONE`
Bootstrap property	No
Required property	No
Restart required	No

Local configuration file	`AgentConfig.properties`
AM console	Tab: `Application` Title: `Session Attribute Fetch Mode`

Cookie Separator Character

The separator for multiple values of the same attribute when it is set as a cookie.

Property name	`org.forgerock.agents.attribute.cookie.separator`

Aliases	`com.sun.identity.agents.config.attribute.cookie.separator`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.attribute.cookie.separator`
	Introduced in Java Agent 5.6

Function	Attributes
Type	String
Default	|
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Application` Title: `Cookie Separator Character`

Fetch Attribute Date Format

The `java.text.SimpleDateFormat` of date attribute values used when an attribute is set in an HTTP header.

Property name	`org.forgerock.agents.attribute.date.format`

Aliases	`com.sun.identity.agents.config.attribute.date.format`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.attribute.date.format`
	Introduced in Java Agent 5.6

Function	Attributes
Type	String
Default	`EEE, d MMM yyyy hh:mm:ss z`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Application` Title: `Fetch Attribute Date Format`

Response Attribute Fetch Mode

Map the name of an AM policy response attribute specified in Response Attribute Map as follows:

- NONE: Do not map
- HTTP_HEADER: Map the to the name of an HTTP response header
- HTTP_COOKIE: Map to the name of an HTTP cookie
- REQUEST_ATTRIBUTE: Map to the name of an HTTP request attribute

Property name	`org.forgerock.agents.response.attribute.fetch.mode`

Aliases	`com.sun.identity.agents.config.response.attribute.fetch.mode`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.response.attribute.fetch.mode`
	Introduced in Java Agent 5.6

Function	Attributes, Response

Supported settings	***NONE*** No AM attributes are mapped. ***HTTP_HEADER*** The named AM attributes are mapped to headers in the outgoing HTTP response. ***REQUEST_ATTRIBUTE*** The named AM attributes are mapped to request attributes in the outgoing HTTP response. ***HTTP_COOKIE*** The named AM attributes are mapped to cookies in the outgoing HTTP response.
Default	`NONE`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Application` Title: `Response Attribute Fetch Mode`

Profile Attribute Fetch Mode

Map the name of an AM profile attribute specified in Profile Attribute Map as follows:

- NONE: Do not map
- HTTP_HEADER: Map the to the name of an HTTP profile header
- HTTP_COOKIE: Map to the name of an HTTP cookie
- REQUEST_ATTRIBUTE: Map to the name of an HTTP profile attribute

Property name	`org.forgerock.agents.profile.attribute.fetch.mode`

Aliases	`com.sun.identity.agents.config.profile.attribute.fetch.mode`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.profile.attribute.fetch.mode`
	Introduced in Java Agent 5.6

Function	Attributes, Cookie reset, Profile

Supported settings	***NONE*** No AM attributes are mapped. ***HTTP_HEADER*** The named AM attributes are mapped to headers in the outgoing HTTP response. ***REQUEST_ATTRIBUTE*** The named AM attributes are mapped to request attributes in the outgoing HTTP response. ***HTTP_COOKIE*** The named AM attributes are mapped to cookies in the outgoing HTTP response.
Default	`NONE`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Application` Title: `Profile Attribute Fetch Mode`

Enable Attribute Encoding

When `true`, attribute values are URL-encoded before being set as a cookie.

Property name	`org.forgerock.agents.attribute.cookie.encode.enabled`

Aliases	`com.sun.identity.agents.config.attribute.cookie.encode`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.attribute.cookie.encode.enabled`
	Introduced in Java Agent 5.6

Function	Attributes
Type	Boolean: `true` returns true; all other strings return `false`.
Default	`true`
Bootstrap property	No
Required property	No

Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Application` Title: `Enable Attribute Encoding` Legacy title: `Attribute Cookie Encode`

Audit

Enable Local Audit Log Rotation

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

Property name	`org.forgerock.agents.local.audit.log.rotation.enabled`

Aliases	`org.forgerock.agents.local.audit.log.rotation.enabled`
	Introduced in Java Agent 5.6
	`com.sun.identity.agents.config.local.log.rotate`
	Introduced in Java Agent 5.0
	Recognized from AM 6

Function	Audit
Type	Boolean: `true` returns true; all other strings return `false`.
Default	`false`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Global` Title: `Enable Local Audit Log Rotation` Legacy title: `Rotate Local Audit Log`

Local Audit Log Rotation Size

The maximum size in bytes of the local audit log files. When Enable Local Audit Log Rotation is `true`, the agent rotates the log file when it reaches this size.

Property name	`org.forgerock.agents.local.audit.log.rotation.bytes`

Aliases	`com.sun.identity.agents.config.local.log.size`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.local.audit.log.rotation.bytes`
	Introduced in Java Agent 5.7

Function	Audit
Type	Integer
Default	`52428800`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Global` Title: `Local Audit Log Rotation Size`

Audit Logfile Retention Count

The number of audit log files to retain after rotation. When the specified limit is reached, the oldest file is deleted when a file rotation occurs.

When the value is `-1`, all rotated files are kept. When the value is, for example, `10`, the current file and nine older rotated files are kept.

Property name	`org.forgerock.agents.local.audit.log.retention.count`
Aliases	`org.forgerock.agents.local.audit.log.retention.count` Introduced in Java Agent 5.7 Recognized from AM 7
Function	Audit
Type	Integer
Default	`-1`
Bootstrap property	No
Required property	No

Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Global (from AM 7)` Title: `Audit Logfile Retention Count`

Audit Access Types

The type of messages to audit.

Property name	`org.forgerock.agents.audit.what`

Aliases	`com.sun.identity.agents.config.audit.accesstype`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.audit.what`
	Introduced in Java Agent 5.6

Function	Audit
Supported settings	***LOG_NONE*** Don't audit anything. ***LOG_ALLOW*** Audit only allowed requests. ***LOG_DENY*** Audit only denied requests. ***LOG_BOTH*** Audit both allowed and denied requests.
Default	`LOG_NONE`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Global` Title: `Audit Access Types`

Local Audit Log Filename

The full path to the agent's local audit log file.

Default: None; local auditing is disabled

Property name	`org.forgerock.agents.local.audit.file.path`

Aliases	`com.sun.identity.agents.config.local.logfile`
	Introduced in Java Agent 5.0
	Recognized from AM 7
	`org.forgerock.agents.local.audit.file.path`
	Introduced in Java Agent 5.6

Function	Audit
Type	String
Bootstrap property	Yes
Required property	No
Restart required	No
Local configuration file	`AgentBootstrap.properties`

Audit Log Location

The location where the agent logs audit messages. If Audit Access Types is `LOG_NONE`, this property has no effect.

Property name	`org.forgerock.agents.audit.where`

Aliases	`com.sun.identity.agents.config.log.disposition`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.audit.where`
	Introduced in Java Agent 5.6

Function	Audit

Supported settings	***NONE*** Don't audit anything, anywhere. ***LOCAL*** Audit locally only. ***REMOTE*** Audit remotely only. ***ALL*** Audit both locally and remotely.
Default	`NONE`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Global` Title: `Audit Log Location`

Authentication failure

Authentication Fail Reason Parameter Name

A query parameter name to contain the reason why authentication failed. The agent appends this parameter to the URL or URI defined by Authentication Fail URL.

If this property is not set, the agent does not append the reason for the authentication failure, when redirecting to the URL or URI.

To reduce the risk of leaking useful information, configure Authentication Fail Reason Parameter Value Map to change the strings for the above values.

Property name	`org.forgerock.agents.authn.fail.reason.parameter.name`
Aliases	`org.forgerock.agents.authn.fail.reason.parameter.name` Introduced in Java Agent 5.7 Recognized from AM 7
Function	Authentication failure
Type	String
Bootstrap property	No

Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Miscellaneous (from AM 7)` Title: `Authentication Fail Reason Parameter Name`

Authentication Fail Reason Parameter Value Map

After an authentication failure, malicious users can use the information you expose to gain access to the system. Map the reason for authentication failure to something generic, or something that is meaningful inside your organization.

When Authentication Fail URL is set, this property maps reasons for authentication failure to custom messages, as follows:

- `AUTHN_BOOKKEEPING_COOKIE_MISSING`: The agent cannot find the authentication tracking cookie. This error can happen if the user successfully authenticates, but clicks the back button of the browser to return to the previous page.
- `NONCE_MISSING`: The agent found the authentication tracking cookie, but it cannot find the unique identifier of the authentication request inside the cookie. This error can happen if the user successfully authenticates, but clicks the back button of the browser to return to the previous page.
- `BAD_AUDIENCE`: The audience in the JWT did not correspond to the audience in the cookie entry. This error can happen if all agents working in a cluster do not have the same Agent Profile Name.
- `NO_TOKEN`: The agent cannot find the session ID token.
- `TOKEN_EXPIRED`: The agent found the session ID token, but it is past its expiry date.
- `AM_SAYS_INVALID`: The agent found the session ID token, the expiry time is correct, but AM returns that the ID token is invalid.
- `JWT_INVALID`: The agent found the session ID token, but cannot parse it.
- `EXCEPTION`: The agent found the session ID token, but threw an exception while parsing it. Alternatively, the agent cannot connect to AM to validate the ID token, maybe due to a network outage.

Specify the authentication failure reason from the preceding table as the map key, and your custom error identifier string as the value. For example:

```
org.forgerock.agents.authn.fail.reason.remapper[TOKEN_EXPIRED]=MY_ERROR_MESSAGE
```

Consider remapping all the failure reasons to a new error message, then be specific on those that hold more meaning for your environment. For example:

```
org.forgerock.agents.authn.fail.reason.remapper=ERROR
```

```
org.forgerock.agents.authn.fail.reason.remapper[AUTHN_BOOKKEEPING_COOKIE_MISSING]=BACK_BUTTON_PRESSED
```

```
org.forgerock.agents.authn.fail.reason.remapper[NONCE_MISSING]=BACK_BUTTON_PRESSED
```

To map all the authentication failure reasons to the same message, you do not need to specify a key in the property.

Property name	`org.forgerock.agents.authn.fail.reason.remapper`
Aliases	`org.forgerock.agents.authn.fail.reason.remapper` Introduced in Java Agent 5.7 Recognized from AM 7
Function	Authentication failure
Type	Map • Keys: failed auth reason code • Values: masked value
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Miscellaneous (from AM 7)` Title: `Authentication Fail Reason Parameter Value Map`

Authentication Fail URL

The URL or URI to which the agent redirects the user after a failed authentication.

If this property is not set, the agent redirects the user to the URL defined in Goto URL. If neither property is set, the agent returns an HTTP 400 Bad Request.

To configure the agent to send the reason for authentication failure in a named query parameter, configure Authentication Fail Reason Parameter Name.

Property name	`org.forgerock.agents.authn.fail.url`
Aliases	`org.forgerock.agents.authn.fail.url` Introduced in Java Agent 5.7 Recognized from AM 7
Function	Authentication failure
Type	String
Bootstrap property	No
Required property	No

Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Miscellaneous (from AM 7)` Title: `Authentication Fail URL` Legacy title: `Authentication Fail Reason Url`

Goto URL

When Authentication Fail URL is not set, this property sets the URL or URI to which the agent redirects the user after a failed authentication. If neither property is set, the agent returns an HTTP 400 Bad Request.

To configure the agent to send the reason for authentication failure in a named query parameter, configure Authentication Fail Reason Parameter Name.

Property name	`org.forgerock.agents.default.goto.url`

Aliases	`com.sun.identity.agents.config.openam.agent.default_goto_url`
	Introduced in Java Agent 5.0
	Recognized from AM 7
	`org.forgerock.agents.default.goto.url`
	Introduced in Java Agent 5.7

Function	Authentication failure
Type	String
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Miscellaneous (from AM 7)` Title: `Goto URL`

Authentication service

AM Authentication Service Path

The path to the AM server.

Property name	`org.forgerock.agents.am.path`

Aliases	`com.iplanet.am.services.deploymentDescriptor`
	Introduced in Java Agent 5.0
	`org.forgerock.agents.am.path`
	Introduced in Java Agent 5.6

Function	Authentication service, Required
Type	String
Bootstrap property	Yes
Required property	Yes - If this property is missing, the agent fails to start
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentBootstrap.properties`

AM Authentication Service Protocol

The protocol used by the AM server. Set to one of the following values:

- HTTP
- HTTPS

Property name	`org.forgerock.agents.am.protocol`

Aliases	`org.forgerock.agents.am.protocol`
	Introduced in Java Agent 5.6
	`com.iplanet.am.server.protocol`
	Introduced in Java Agent 5.0
	Recognized from AM 6

Function	Authentication service, Required
Type	String
Bootstrap property	Yes
Required property	Yes - If this property is missing, the agent fails to start
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentBootstrap.properties`

AM console	Tab: `AM Services` Title: `AM Authentication Service Protocol`

Encryption Key/Salt

The key/salt used by the default encryption class to perform encryption and decryption within the agent.

Generate a secure, random key with `agentadmin --key` or `agentadmin --rotate` commands as described in Rotate the agent profile password.

If this property is not set, the agent terminates with a configuration error.

> **Warning**
>
> If you are changing this property manually or with `agentadmin --key`, you must re-encrypt the agent profile password. For more information, refer to Encrypted Agent Password.

Property name	`org.forgerock.agents.encryption.key`

Aliases	`org.forgerock.agents.encryption.key`
	Introduced in Java Agent 5.10.1
	Recognized from AM 7.1
	`am.encryption.pwd`
	Introduced in Java Agent 5.0
	`com.sun.identity.client.encryptionKey`
	Introduced in Java Agent 5.0

Function	Authentication service, Encryption
Type	String
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`

AM Authentication Service Host Name

The AM server host name.

Property name	`org.forgerock.agents.am.hostname`

Aliases	`com.iplanet.am.server.host`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.am.hostname`
	Introduced in Java Agent 5.6

Function	Authentication service, Required
Type	String
Bootstrap property	Yes
Required property	Yes - If this property is missing, the agent fails to start
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentBootstrap.properties`
AM console	Tab: `AM Services` Title: `AM Authentication Service Host Name`

AM Authentication Service Port

The AM server port number.

Property name	`org.forgerock.agents.am.port`

Aliases	`com.iplanet.am.server.port`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.am.port`
	Introduced in Java Agent 5.6

Function	Authentication service, Required
Type	String
Bootstrap property	Yes
Required property	Yes - If this property is missing, the agent fails to start
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentBootstrap.properties`
AM console	Tab: `AM Services` Title: `AM Authentication Service Port`

Bad configuration detection

Bad advice loop termination HTTP status

When a user has insufficient credentials to access a requested resource, AM can return policy advice that requires the user to authenticate at a higher level.

If there is an error in the AM configuration, an infinite authentication loop can occur, where the user is repeatedly asked to authenticate.

This property defines the HTTP status code to return to the user's browser when the agent breaks a loop.

Integers in the range 100-299 and 400-599 are valid; however, not all browsers accept all status codes in this range.

Property name	`org.forgerock.agents.bad.advice.loop.termination.http.code`
Aliases	`org.forgerock.agents.bad.advice.loop.termination.http.code` Introduced in Java Agent 2023.11 Recognized from AM 7.1
Function	Bad configuration detection
Type	Integer
Default	`508`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`

Bad advice loop termination counter

When a user has insufficient credentials to access a requested resource, AM can return policy advice that requires the user to authenticate at a higher level.

If there is an error in the AM configuration, an infinite authentication loop can occur, where the user is repeatedly asked to authenticate.

This property defines the maximum number of times the agent asks the user to authenticate before breaking a loop.

Property name	`org.forgerock.agents.bad.advice.loop.termination.counter`

Aliases	`org.forgerock.agents.bad.advice.loop.termination.counter` Introduced in Java Agent 2023.11 Recognized from AM 7.1
Function	Bad configuration detection
Type	Integer
Default	`5`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`

Bad advice loop termination URL

When a user has insufficient credentials to access a requested resource, AM can return policy advice that requires the user to authenticate at a higher level.

If there is an error in the AM configuration, an infinite authentication loop can occur, where the user is repeatedly asked to authenticate.

This property defines the URL or URI of a page to display when the agent breaks a loop. Consider indicating on the page that there is an AM configuration error.

Property name	`org.forgerock.agents.bad.advice.loop.termination.url`
Aliases	`org.forgerock.agents.bad.advice.loop.termination.url` Introduced in Java Agent 2023.11 Recognized from AM 7.1
Function	Bad configuration detection
Type	String
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`

Client identification

Client Hostname Header

The name of the HTTP header used to determine the hostname of a client. See also Client IP Address Header.

If this property is not set, the value returned by `HttpServletRequest.getRemoteHost` is used.

Property name	`org.forgerock.agents.http.header.containing.remote.hostname`

Aliases	`org.forgerock.agents.http.header.containing.remote.hostname`
	Introduced in Java Agent 5.6
	`com.sun.identity.agents.config.client.hostname.header`
	Introduced in Java Agent 5.0
	Recognized from AM 6

Function	Client identification, Continuous security
Type	String
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Advanced` Title: `Client Hostname Header`

Client IP Validation Mode

For each authenticated request from a named web application, check that the IP address of the request satisfies one of the following acceptance criteria:

- It originates from the IP address used for first authentication.
- It has acceptable changes only, as mapped in Client IP Validation Address Map
- If the web application is not named, check the the IP address globally, for all web applications.

Property name	`org.forgerock.agents.original.ip.check.mode.map`
Aliases	`org.forgerock.agents.original.ip.check.mode.map` Introduced in Java Agent 5.8.0 Recognized from AM 7.1

Function	Client identification
Supported settings	***OFF*** IP address checking is disabled. ***DENY*** An "unacceptable" IP address change triggers an HTTP 403 response. ***LOGOUT*** An "unacceptable" IP address change causes the agent to invalidate the user token by calling the logout endpoint in AM and killing the user's cookies.
Default	`OFF`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Application (from AM 7.1)` Title: `Client IP Validation Mode`

Client IP Validation Address Map

A map of acceptable alternative values for IP addresses, or address ranges in CIDR format, that incoming requests may change to without triggering DENY or LOGOUT behaviour.

This property is used by Client IP Validation Mode.

Property name	`org.forgerock.agents.acceptable.ip.address.map`
Aliases	`org.forgerock.agents.acceptable.ip.address.map` Introduced in Java Agent 5.8.0 Recognized from AM 7.1
Function	Client identification
Type	Map • Keys: web application • Values: acceptable IP addresses, comma separated, CIDR format acceptable
Bootstrap property	No

Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Application (from AM 7.1)` Title: `Client IP Validation Address Map`

Client IP Address Header

The name of the HTTP header used to determine the IP address of a client. See also Client Hostname Header.

If this property is not set, the value returned by `HttpServletRequest.getRemoteAddr` is used.

Property name	`org.forgerock.agents.http.header.containing.ip.address`

Aliases	`com.sun.identity.agents.config.client.ip.header`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.http.header.containing.ip.address`
	Introduced in Java Agent 5.6

Function	Client identification, Continuous security
Type	String
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Advanced` Title: `Client IP Address Header`

Configure behaviour

Retain previous override behaviour

When `true`, the agent does not substitute the Alternative Agent Host Name, Alternative Agent Port Number or Alternative Agent Protocol into the URL used for matching against not enforced rules, or policy evaluation.

When `false`, the values are substituted as expected.

The default value preserves backward compatibility.

Property name	`org.forgerock.agents.retain.previous.override.behaviour.enabled`
Aliases	`org.forgerock.agents.retain.previous.override.behaviour.enabled` Introduced in Java Agent 2023.6
Function	Configure behaviour
Type	Boolean: `true` returns true; all other strings return `false`.
Default	`true`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`

Connection pooling

Max HTTP Connection Count

When Enable Connection Pooling is `true`, this property defines the maximum number of HTTP connections allowed at any time.

Property name	`org.forgerock.agents.http.client.max.connections`
Aliases	`org.forgerock.agents.http.client.max.connections` Introduced in Java Agent 5.8.0
Function	Connection pooling
Type	Integer
Default	`1000`
Bootstrap property	Yes
Required property	No
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentBootstrap.properties`

AM console	Tab: `Redirect` Title: `Max HTTP Connection Count`

HTTP Connection Timeout

When Enable Connection Pooling is `true`, this property defines the connection timeout in seconds.

Property name	`org.forgerock.agents.http.client.connection.timeout.seconds`
Aliases	`org.forgerock.agents.http.client.connection.timeout.seconds` Introduced in Java Agent 5.8.0
Function	Connection pooling
Type	Integer
Default	`10`
Bootstrap property	Yes
Required property	No
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentBootstrap.properties`
AM console	Tab: `Redirect` Title: `HTTP Connection Timeout`

Enable HTTP Connection Reuse

When Enable Connection Pooling is `true`, this property enables connection reuse.

Property name	`org.forgerock.agents.http.client.reuse.connections.enabled`
Aliases	`org.forgerock.agents.http.client.reuse.connections.enabled` Introduced in Java Agent 5.8.0
Function	Connection pooling
Type	Boolean: `true` returns true; all other strings return `false`.
Default	`true`
Bootstrap property	Yes

Required property	No
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentBootstrap.properties`

Enable HTTP Connection State

This option only applies when these properties are `true`:

- Enable Connection Pooling
- Enable HTTP Connection Reuse

Set this property to `true` to change the Apache HTTP Client default behavior, and allow connection reuse.

Because the client certificate is defined at the client level, all requests to the same target share the same client certificate, so enabling this property should not be an issue.

Property name	`org.forgerock.agents.http.client.connection.state.enabled`
Aliases	`org.forgerock.agents.http.client.connection.state.enabled` Introduced in Java Agent 5.8.0
Function	Connection pooling
Type	Boolean: `true` returns true; all other strings return `false`.
Default	`true`
Bootstrap property	Yes
Required property	No
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentBootstrap.properties`

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.

Property name	`org.forgerock.agents.use.connection.pooling.enabled`

Aliases	`org.forgerock.agents.use.connection.pooling.enabled` Introduced in Java Agent 5.8.0
Function	Connection pooling
Type	Boolean: `true` returns true; all other strings return `false`.
Default	`false`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`

HTTP Socket Timeout

When Enable Connection Pooling is `true`, this property defines the socket timeout in seconds.

Property name	`org.forgerock.agents.http.client.socket.timeout.seconds`
Aliases	`org.forgerock.agents.http.client.socket.timeout.seconds` Introduced in Java Agent 5.8.0
Function	Connection pooling
Type	Integer
Default	`10`
Bootstrap property	Yes
Required property	No
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentBootstrap.properties`

Enable HTTP Retry

When Enable Connection Pooling is `true`, this property enables retries after failed requests.

Property name	`org.forgerock.agents.http.client.retry.requests.enabled`

Aliases	`org.forgerock.agents.http.client.retry.requests.enabled` Introduced in Java Agent 5.8.0
Function	Connection pooling
Type	Boolean: `true` returns true; all other strings return `false`.
Default	`true`
Bootstrap property	Yes
Required property	No
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentBootstrap.properties`

Container

Container Character Encoding

The character encoding used by the agent when encoding extended characters in the resource paths of not-enforced rules.

Property name	`org.forgerock.agents.container.encoding`
Aliases	`org.forgerock.agents.container.encoding` Introduced in Java Agent 5.9.1
Function	Container, Not-enforced
Type	String
Default	`UTF-8`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`

Container Parameter Encoding

The character encoding used by the agent when encoding extended characters in the HTTP query parameters of not-enforced rules.

Property name	`org.forgerock.agents.container.param.encoding`
Aliases	`org.forgerock.agents.container.param.encoding` Introduced in Java Agent 5.9.1
Function	Container, Not-enforced
Type	String
Default	`ISO-8859-1`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`

Continuous security

Continuous Security Cookie Map

Maps cookie values available in inbound resource requests to entries in the environmental conditions map, which agents send to AM during policy evaluation.

Property name	`org.forgerock.agents.continuous.security.cookies.map`

Aliases	`org.forgerock.agents.continuous.security.cookies.map`
	Introduced in Java Agent 5.6
	`org.forgerock.openam.agents.config.continuous.security.cookies`
	Introduced in Java Agent 5.0
	Recognized from AM 6

Function	Continuous security
Type	Map • Keys: incoming cookie name • Values: name of entry in environment map
Bootstrap property	No
Required property	No
Restart required	No

Local configuration file	`AgentConfig.properties`
AM console	Tab: `Application` Title: `Continuous Security Cookie Map` Legacy title: `Continuous Security Cookies`

Continuous Security Header Map

Maps header values in inbound resource requests to entries in the environmental conditions map, which agents send to AM during policy evaluation.

Example:

`org.forgerock.agents.continuous.security.headers.map[User-Agent]=myUserAgentHeaderEntry`

Property name	`org.forgerock.agents.continuous.security.headers.map`

Aliases	`org.forgerock.agents.continuous.security.headers.map`
	Introduced in Java Agent 5.6
	`org.forgerock.openam.agents.config.continuous.security.headers`
	Introduced in Java Agent 5.0
	Recognized from AM 6

Function	Continuous security
Type	Map • Keys: incoming header name • Values: name of entry in environment map
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Application` Title: `Continuous Security Header Map` Legacy title: `Continuous Security Headers`

Client Hostname Header

The name of the HTTP header used to determine the hostname of a client. See also Client IP Address Header.

If this property is not set, the value returned by `HttpServletRequest.getRemoteHost` is used.

Property name	`org.forgerock.agents.http.header.containing.remote.hostname`

Aliases	`org.forgerock.agents.http.header.containing.remote.hostname`
	Introduced in Java Agent 5.6
	`com.sun.identity.agents.config.client.hostname.header`
	Introduced in Java Agent 5.0
	Recognized from AM 6

Function	Client identification, Continuous security
Type	String
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Advanced` Title: `Client Hostname Header`

Client IP Address Header

The name of the HTTP header used to determine the IP address of a client. See also Client Hostname Header.

If this property is not set, the value returned by `HttpServletRequest.getRemoteAddr` is used.

Property name	`org.forgerock.agents.http.header.containing.ip.address`

Aliases	`com.sun.identity.agents.config.client.ip.header`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.http.header.containing.ip.address`
	Introduced in Java Agent 5.6

Function	Client identification, Continuous security
Type	String
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`

AM console	Tab: `Advanced` Title: `Client IP Address Header`

Cookie

Maximum Decompression Size

The maximum size, in bytes, a compressed cookie is allowed to expand to when decompressed.

Property name	`org.forgerock.agents.max.decompression.size.bytes`
Aliases	`org.forgerock.agents.max.decompression.size.bytes` Introduced in Java Agent 5.10.0
Function	Cookie
Type	Integer
Default	`32768`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`

Pre-Authn and Post Data Preservation Cookie Signing Value

The key to sign pre-authentication cookies and POST data preservation cookies.

The key is set during installation, when the agent requests the path to a file containing the cookie signing key, and then uses the key to set the cookie signing value in the `AgentKey.properties` file. For information about how to set or change the key after installation, see Rotate cookie signing keys.

For security, you are recommended to configure cookie signing. The agent does not sign cookies when:

- The path to the signing key is left blank during installation.
- The signing key in the `AgentKey.properties` file is less than 64-characters long.

Property name	`org.forgerock.agents.cookie.signing.value`
Aliases	`org.forgerock.agents.cookie.signing.value` Introduced in Java Agent 5.10.0

Function	Cookie, POST data preservation, Pre-authentication
Type	String
Bootstrap property	No
Required property	No
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentKey.properties`

Max Age of Pre-Authentication Cookie

The maximum age in seconds of the pre-authentication cookie configured in Pre-Authentication Cookie Name.

Property name	`org.forgerock.agents.authn.cookie.max.age.seconds`
Aliases	`org.forgerock.agents.authn.cookie.max.age.seconds` Introduced in Java Agent 5.6.3.0 Recognized from AM 7
Function	Cookie, Pre-authentication
Type	Integer
Default	`600`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Global (from AM 7)` Title: `Max Age of Pre-Authentication Cookie` Legacy title: `Pre-Authenticated Cookie Max Age`

Load Balancer Cookie Name

The load balancer cookie name. Make sure that this property has the same value as the AM property `com.iplanet.am.lbcookie.name`.

Property name	`org.forgerock.agents.load.balancer.cookie.name`
Aliases	`org.forgerock.agents.load.balancer.cookie.name` Introduced in Java Agent 5.8.0 Recognized from AM 7.1
Function	Cookie
Type	String
Default	`amlbcookie`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Global (from AM 7.1)` Title: `Load Balancer Cookie Name`

Enable Encoded Cookies

When `true`, cookies are base64-encoded.

Property name	`com.iplanet.am.cookie.encode`
Aliases	`com.iplanet.am.cookie.encode` Introduced in Java Agent 5.6 Recognized from AM 7
Function	Cookie
Type	Boolean: `true` returns true; all other strings return `false`.
Default	`false`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`

AM console	Tab: `SSO (from AM 7)` Title: `Enable Encoded Cookies` Legacy title: `Encode Cookies`

Enable HTTP Only Cookies

When `true`, cookies are flagged as `HTTPOnly`. Use this property to prevent scripts and third-party programs from accessing the cookies.

Property name	`com.sun.identity.cookie.httponly`
Aliases	`com.sun.identity.cookie.httponly` Introduced in Java Agent 5.0 Recognized from AM 7
Function	Cookie
Type	Boolean: `true` returns true; all other strings return `false`.
Default	`true`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `SSO (from AM 7)` Title: `Enable HTTP Only Cookies` Legacy title: `Http Only`

Pre-Authentication Cookie Name

The name of the pre-authentication cookie. This cookie tracks the progress of authentication with AM, and protects requests from replay attacks. It contains the following information:

- URL of the original request
- HTTP mode
- Secure ID (subsequently baked into the nonce of the returned JWT)
- Relevant ACR information
- Transaction ID

- Expiry time configured by Max Age of Pre-Authentication Cookie

(Before Java Agent 5.7), 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.

(Java Agent 5.7 and later versions) The agent can optionally create a cookie for each authentication request to AM. In some environments, this creates 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 can limit how many cookies they handle.

Configure the cookie name as follows:

- To use one cookie for all concurrent authentication requests to AM, configure as a string. For example, `org.forgerock.agents.authn.cookie.name=cookie-name`.
- To use one cookie for each authentication request to AM, configure as `%n`, or as `%n` before, in the middle of, or after a string. When the agent creates the cookie, it translates the string `%n` into a unique identifier. For example:
  - `org.forgerock.agents.authn.cookie.name=%n`
  - `org.forgerock.agents.authn.cookie.name=%n-cookie-name`
  - `org.forgerock.agents.authn.cookie.name=cookie-%n-name`
  - `org.forgerock.agents.authn.cookie.name=cookie-name-%n`

The agent compresses and then signs the cookie.

Property name	`org.forgerock.agents.authn.cookie.name`

Aliases	`com.sun.identity.agents.config.cdsso.cookie.name`
	Introduced in Java Agent 5.0
	Recognized from AM 7
	`org.forgerock.agents.authn.cookie.name`
	Introduced in Java Agent 5.6

Function	Cookie, Pre-authentication
Type	String
Default	`amFilterCDSSORequest`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`

AM console	Tab: `Global (from AM 7)` Title: `Pre-Authentication Cookie Name` Legacy title: `Pre-Authenticated Cookie Name`

Post Data Preservation Cookie Name

The name of the Post Data Preservation cookie. This cookie maintains the security of the data in unauthenticated POST requests. It contains a unique one-time code which is cross-checked against the stored data making it extremely difficult for malicious actors to replay the stored data for other users.

Since Java Agent 5.10, there is the option of creating one cookie for all concurrent PDP requests, or alternatively to have one uniqely named cookie per request.

If you have tests in your environment that make multiple PDP requests to the agent, you may find intermittent failures as browsers can limit how many cookies they handle.

Configure the cookie name as follows:

- To use one cookie for all concurrent PDP requests to AM, configure as a string. For example, `org.forgerock.agents.pdp.cookie.name=cookie-name`.
- To use one cookie for each authentication request to AM, configure as `%n` before, in the middle, or after a string. When the agent creates the cookie, it substitutes `%n` for a unique identifier. For example:
  - `org.forgerock.agents.pdp.cookie.name=%n`
  - `org.forgerock.agents.pdp.cookie.name=%n-cookie-name`
  - `org.forgerock.agents.pdp.cookie.name=cookie-%n-name`
  - `org.forgerock.agents.pdp.cookie.name=cookie-name-%n`

The agent compresses and then signs the cookie.

Property name	`org.forgerock.agents.pdp.cookie.name`
Aliases	`org.forgerock.agents.pdp.cookie.name` Introduced in Java Agent 5.10.0 Recognized from AM 7.1
Function	Cookie, POST data preservation
Type	String
Default	`PDP_Nonce`
Bootstrap property	No
Required property	No

Restart required	No
Local configuration file	`AgentConfig.properties`

Enable Load Balancer Cookies

When `true`, the agent writes load balancer cookies each time it invokes AM.

Use this property with Load Balancer Cookie Name to improve performance. Load balancer cookies can reduce the number of calls that different AM instances make to the Core Token Service (CTS).

Because the agent invokes AM to log out a user, it creates a load-balancer cookie on logout. To remove these cookies on logout, add them to the Reset Cookie List.

Property name	`org.forgerock.agents.load.balancer.cookies.enabled`
Aliases	`org.forgerock.agents.load.balancer.cookies.enabled` Introduced in Java Agent 5.8.0 Recognized from AM 7.1
Function	Cookie
Type	Boolean: `true` returns true; all other strings return `false`.
Default	`false`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Global (from AM 7.1)` Title: `Enable Load Balancer Cookies` Legacy title: `Load Balancer Cookie Enabled`

Cookie reset

Reset Cookie List

A list of cookies to reset before redirecting the client for login, and when the client logs out. Cookie Reset must be `true`.

The agent searches for the cookie name using a case-sensitive search. If it finds a match, the cookie is reset. Otherwise, the agent searches again using a case-insensitive search. If it then finds a match, the cookie is reset and a warning is issued to the logs.

The list does not need to include the names of cookies created when Profile Attribute Fetch Mode or Session Attribute Fetch Mode has the value `HTTP_COOKIE`.

Property name	`org.forgerock.agents.cookie.reset.name.list`

Aliases	`org.forgerock.agents.cookie.reset.name.list`
	Introduced in Java Agent 5.6
	`com.sun.identity.agents.config.cookie.reset.name`
	Introduced in Java Agent 5.0
	Recognized from AM 6

Function	Cookie reset
Type	List
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `SSO` Title: `Reset Cookie List` Legacy title: `Cookies Reset Name List`

Session Attribute Fetch Mode

Map the name of an AM session attribute specified in Session Attribute Map as follows:

- NONE: Do not map
- HTTP_HEADER: Map the to the name of an HTTP session header
- HTTP_COOKIE: Map to the name of an HTTP cookie
- REQUEST_ATTRIBUTE: Map to the name of an HTTP session attribute

When the value is `HTTP_COOKIE`, Cookie Reset is automatically set to `true`. Before redirecting the client for login, and when the client logs out, the agent resets profile and session attributes cookies, and cookies in the Reset Cookie List.

Property name	`org.forgerock.agents.session.attribute.fetch.mode`

Aliases	`org.forgerock.agents.session.attribute.fetch.mode`
	Introduced in Java Agent 5.6
	`com.sun.identity.agents.config.session.attribute.fetch.mode`
	Introduced in Java Agent 5.0
	Recognized from AM 6

Function	Attributes, Cookie reset, Session
Supported settings	***NONE*** No AM attributes are mapped. ***HTTP_HEADER*** The named AM attributes are mapped to headers in the outgoing HTTP response. ***REQUEST_ATTRIBUTE*** The named AM attributes are mapped to request attributes in the outgoing HTTP response. ***HTTP_COOKIE*** The named AM attributes are mapped to cookies in the outgoing HTTP response.
Default	`NONE`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Application` Title: `Session Attribute Fetch Mode`

Cookie Reset

When `true`, the agent resets the cookies in the response before redirecting the client for login, and when the client logs out.

The agent resets the cookies listed in Reset Cookie List, and cookies that store profile or session attributes (when Profile Attribute Fetch Mode or Session Attribute Fetch Mode has the value `HTTP_COOKIE`).

To reset cookies that store response attributes (when Response Attribute Fetch Mode has the value `HTTP_COOKIE`), add them to the Reset Cookie List.

Property name	`org.forgerock.agents.cookie.reset.enabled`

Aliases	`org.forgerock.agents.cookie.reset.enabled`
	Introduced in Java Agent 5.6
	`com.sun.identity.agents.config.cookie.reset.enable`
	Introduced in Java Agent 5.0
	Recognized from AM 6

Function	Cookie reset

Type	Boolean: `true` returns true; all other strings return `false`.
Default	`false`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `SSO` Title: `Cookie Reset`

Reset Cookie Domain Map

Maps the cookie name specified in Reset Cookie List to a domain. After the cookie reset, the cookie is used in the domain.

Property name	`org.forgerock.agents.cookie.reset.domain.map`

Aliases	`org.forgerock.agents.cookie.reset.domain.map`
	Introduced in Java Agent 5.6
	`com.sun.identity.agents.config.cookie.reset.domain`
	Introduced in Java Agent 5.0
	Recognized from AM 6

Function	Cookie reset
Type	Map • Keys: cookie name • Values: cookie domain
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `SSO` Title: `Reset Cookie Domain Map` Legacy title: `Cookies Reset Domain Map`

Reset Cookie Path Map

Maps cookie name specified in Reset Cookie List to a path. After cookie reset, the cookie is used in the path.

Property name	`org.forgerock.agents.cookie.reset.path.map`

Aliases	`com.sun.identity.agents.config.cookie.reset.path`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.cookie.reset.path.map`
	Introduced in Java Agent 5.6

Function	Cookie reset
Type	Map • Keys: cookie name • Values: cookie path
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `SSO` Title: `Reset Cookie Path Map` Legacy title: `Cookies Reset Path Map`

Profile Attribute Fetch Mode

Map the name of an AM profile attribute specified in Profile Attribute Map as follows:

- NONE: Do not map
- HTTP_HEADER: Map the to the name of an HTTP profile header
- HTTP_COOKIE: Map to the name of an HTTP cookie
- REQUEST_ATTRIBUTE: Map to the name of an HTTP profile attribute

Property name	`org.forgerock.agents.profile.attribute.fetch.mode`

Aliases	`com.sun.identity.agents.config.profile.attribute.fetch.mode`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.profile.attribute.fetch.mode`
	Introduced in Java Agent 5.6

Function	Attributes, Cookie reset, Profile
Supported settings	***NONE*** No AM attributes are mapped. ***HTTP_HEADER*** The named AM attributes are mapped to headers in the outgoing HTTP response. ***REQUEST_ATTRIBUTE*** The named AM attributes are mapped to request attributes in the outgoing HTTP response. ***HTTP_COOKIE*** The named AM attributes are mapped to cookies in the outgoing HTTP response.
Default	`NONE`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Application` Title: `Profile Attribute Fetch Mode`

Cross-domain single sign-on

Transmit Cookies Securely

When `true`, all cookies written by the agent are secure. For backward compatibility, the default is `false`.

Property name	`org.forgerock.agents.secure.cookies.enabled`

Aliases	`com.sun.identity.agents.config.cdsso.secure.enable`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.secure.cookies.enabled`
	Introduced in Java Agent 5.6
	`com.iplanet.am.cookie.secure`
	Introduced in Java Agent 5.0

Function	Cross-domain single sign-on
Type	Boolean: `true` returns true; all other strings return `false`.
Default	`false`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `SSO` Title: `Transmit Cookies Securely` Legacy title: `CDSSO Secure Enable`

Authentication Redirect URI

The URI the agent uses to process authentication requests.

When this property is not defined, the redirect URI is provided by AM.

When this property is defined and Location of Agent Configuration Repository is `REMOTE`, AM overwrites this property.

If OIDC authentication is being used, changing the value of this property while the agent is running prevents it from functioning. Restart the agent immediately after the value in AM is altered and the properties saved.

Property name	`org.forgerock.agents.authn.redirect.uri`

Aliases	`org.forgerock.agents.authn.redirect.uri`
	Introduced in Java Agent 5.6
	`com.sun.identity.agents.config.cdsso.redirect.uri`
	Introduced in Java Agent 5.0
	Recognized from AM 6

Function	Cross-domain single sign-on, Required
Type	String

Bootstrap property	No
Required property	No
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentConfig.properties`
AM console	Tab: `SSO` Title: `Authentication Redirect URI` Legacy title: `CDSSO Redirect URI`

Cross-site scripting

XSS Code Element List

Strings that, when found in the request, cause the agent to redirect the client to an error page.

Property name	`org.forgerock.agents.xss.code.element.list`

Aliases	`org.forgerock.agents.xss.code.element.list`
	Introduced in Java Agent 5.7
	`com.sun.identity.agents.config.xss.code.elements`
	Introduced in Java Agent 5.0
	Recognized from AM 6

Function	Cross-site scripting
Type	List
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Advanced` Title: `XSS Code Element List` Legacy title: `Possible XSS code elements`

XSS Redirect URI Map

A map of web application name to URI. When a cross-site scripting attack is detected, the agent redirects to the URI specified in the map. The URI is expected to be a page (HTML, or otherwise) indicating that such an attack has been detected.

For example, to redirect clients of MyApp to `/myapp/error.html`, enter MyApp as the map key and `/myapp/error.html` as the map value.

Property name	`org.forgerock.agents.xss.redirect.uri.map`

Aliases	`com.sun.identity.agents.config.xss.redirect.uri`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.xss.redirect.uri.map`
	Introduced in Java Agent 5.7

Function	Cross-site scripting
Type	Map • Keys: web application • Values: cross site scripting URI
Default	`/agentapp/XSSCodeDetected.html`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Advanced` Title: `XSS Redirect URI Map` Legacy title: `XSS detection redirect URI`

Custom login redirect

Enable SSO Token Acceptance

Set this property as follows:

- `true`: Accept SSO tokens. Use this option when the agent and the token issuer are in the same domain, and for web applications and APIs where the backend accepts user information from SSO tokens.
- `false`: Do not accept SSO tokens; require OIDC JWTs for authentication.

During session upgrade the format of the composite advice is as follows:

- When both this property and Enable Custom Login Mode are `true`, the composite advice has the following format: `?authIndexType=composite_advice&authIndexValue=<Advices Value>`
- When either property is `false`, the composite advice has the following format: `?composite_advice=<Advices Value>`

Property name	`org.forgerock.agents.accept.sso.tokens.enabled`

Aliases	`org.forgerock.agents.accept.sso.tokens.enabled`
	Introduced in Java Agent 5.7.1
	Recognized from AM 7.1
	`org.forgerock.agents.accept.sso.tokens`
	Introduced in Java Agent 5.7.1
	`com.forgerock.agents.accept.sso.tokens`
	Introduced in Java Agent 5.7.1

Function	Custom login redirect, Login redirect, SSO cookie handling
Type	Boolean: `true` returns true; all other strings return `false`.
Default	`false`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `SSO (from AM 7.1)` Title: `Enable SSO Token Acceptance` Legacy title: `Accept SSO Tokens`

OAuth Login URL List

Use only when Enable Custom Login Mode is `false` and AM Login URL List is empty.

Specify rules to evaluate the incoming request URL, based on domain, path, request header, or query parameters. Specify a URL for login redirect with optional parameters.

Format, with no spaces between values:

`[domain/path][header:value][?param=value[,param=value]]|[URL][?param=value&param=value]`

When an unauthenticated request URL matches a rule specified by this property, the agent redirects the request to the specified URL for login.

When this property configures multiple rules, the agent sorts the rules into the following order and applies them in that order until it finds a match:

1. Header specification - a rule with a header specification is applied before other rules
2. Longest domain
3. Longest path

4. Highest number of parameter specifications

During redirect, the agent appends the goto parameter configured in Goto Parameter Name, and a nonce parameter, to the agent's CDSSO endpoint. If Enable FQDN Checking is `true`, the agent iterates through the list of URLs until it finds a redirect URL that matches the FQDN check values. Otherwise, the agent redirects the user to the URL configured in the conditional redirect rules.

`[domain/path]`

The incoming request URL:

- Domain: For example, `example.com`. The agent must match the domain and its subdomains. For example, `example.com` matches `mydomain.example.com` and `www.example.com`. Domains can also include path information, for example, `example.com/market`, but cannot specify ports.
- Subdomain: For example, `mydomain.example.com`. The agent match the domain, the subdomain, and any sub-subdomain. For example, `mydomain.example.com` matches `true.mydomain.example.com`. Subdomains can include path information, for example, `mydomain.example.com/s6ecure`, but cannot specify ports.
- Path: For example, `/myapp`.
- No value: Nothing is specified before the | character and the rule applies to every incoming request.

`[header:value]`

One header/value pair provided in the incoming request. If the header value is not provided, the header can take any value. For example:

Requests containing a header called `X-local` with the value `provided` are redirected to the default URL:

```
org.forgerock.agents.oauth.login.url.list[0]= X-local:provided|
```

Requests containing a header called `X-local` with any value are redirected to the default URL:

```
org.forgerock.agents.oauth.login.url.list[0]= X-local:|
```

`[?param=value[,param=value]`

One or more parameter and value pairs provided in the incoming request. If the parameter value is not provided, the parameter can take any value. For example:

Requests containing a parameter called `site` with the value `shopping` are redirected to the default URL:

```
org.forgerock.agents.oauth.login.url.list[2]= ?site=shopping|
```

Requests containing a parameter called `target`with the value `cooking` **AND** a parameter called `price` with the value `low` are redirected to the default URL:+

```
org.forgerock.openam.agents.config.conditional.login.url[0]= ?target=cooking,price=low|
```

`[URL]`

The login URL. The URL can be an AM instance, an AM site, or a 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/login.jsp
```

`https://am.example.com:8443/openam/XUI/#login/`

`https://am.example.com:8443/openam/customlogin/login.jsp`

If `[URL]` is not specified, the agent redirects the request to the AM instance or site specified by the following bootstrap properties:

`org.forgerock.agents.am.protocol://org.forgerock.agents.am.hostname:org.forgerock.agents.am.port/org.forgerock.agents.am.path`

`[?param=value&param=value]`

One or more parameters to add to the login URL. Chain multiple parameters with an ampersand (&), for example, `realm=value&parameter1=value1&parameter2=value2`.

When the parameter is `?realm=value` it specifies the AM realm into which the agent logs the users. For example, `?realm=marketplace`.

When redirecting to AM's XUI, use an ampersand (&) instead of a question mark (?). For example, `https://am.example.com:8443/openam/XUI/#login/&realm=marketplace`.

A realm parameter is not required in the login URL when any of the following conditions are true:

- The custom login page itself sets the realm parameter, for example, because it lets the user choose it. In this case, you must ensure the custom login page always returns a realm parameter to the agent.
- The realm that the agent is logging the user into has DNS aliases configured in AM.
- AM logs the user into the realm whose DNS alias matches the incoming request URL. For example, an inbound request from the http://marketplace.example.com URL logs in the marketplace realm if the realm alias is set to marketplace.example.com.
- The users should always log in to the Top Level Realm.

Examples

+

Requests containing a header called `X-local` with the value `provided` are redirected to the specified URL in the `beta` realm:

+

`org.forgerock.agents.oauth.login.url.list[0]= X-local:provided|http://mysite.local.com:8081/login?realm=beta`

+

Requests containing a header called `X-local` with any value are redirected to the default URL in the `gamma` realm:

+

`org.forgerock.agents.oauth.login.url.list[1]= X-local:|?realm=gamma`

+

Requests containing a parameter called `site` with the value `shopping` **AND** a parameter called `mode` with the value `discount` are redirected to the default URL in the `discountshopping` realm:

+

`org.forgerock.agents.oauth.login.url.list[2]= ?site=shopping,mode=discount|?realm=discountshopping`

+

Requests containing a parameter called `target` with the value `cooking` are redirected to the AM XUI page in the `kitchen` realm. Note the use of `&` before the realm parameter:

+

`org.forgerock.openam.agents.config.conditional.login.url[0]= ?target=cooking|https://am.example.com:8443/openam/XUI/#login/&realm=kitchen`

+

Requests containing a parameter called `target` with the value `cooking` are redirected to a non-AM login page in the `kitchen` realm. Note the use of `?` before the realm parameter:

+

`org.forgerock.openam.agents.config.conditional.login.url[0]= ?target=cooking|https://mysite.example.com:8443/login/?realm=kitchen`

Property name	`org.forgerock.agents.oauth.login.url.list`

Aliases	`org.forgerock.agents.oauth.login.url.list`
	Introduced in Java Agent 5.6
	`org.forgerock.openam.agents.config.conditional.login.url`
	Introduced in Java Agent 5.6
	Recognized from AM 6

Function	Custom login redirect, Default Login Redirect, Login redirect, Login Redirect (Default)
Type	List
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `AM Services` Title: `OAuth Login URL List` Legacy title: `AM Conditional Login URL`

Enable Custom Login Mode

Set the login redirection mode, as follows:

- `false`: Use default login redirection mode.
  - The agent can redirect requests to any AM instance that supports the `/oauth2/authorize` endpoint. By default, this is the AM instance that is specified during installation.
  - The `/oauth2/authorize` endpoint returns an OIDC ID token. This is the only response the agent accepts.
  - Use with OAuth Login URL List to modify or redirect calls to the endpoint that provides the tokens.
- `true`: Use custom login redirection mode.
  - The agent handles JWTs or SSO tokens as session tokens for authentication and authorization, and can can redirect login anywhere.
  - Use with AM Login URL List and Legacy Login URL List to modify or redirect calls.

During session upgrade, the format of the composite advice is as follows:

- When both this property and Enable SSO Token Acceptance are `true`, the composite advice has the following format: `?authIndexType=composite_advice&authIndexValue=<Advices Value>`
- When either property is `false`, the composite advice has the following format: `?composite_advice=<Advices Value>`

Property name	`org.forgerock.agents.legacy.login.enabled`

Aliases	`org.forgerock.agents.legacy.login.enabled`
	Introduced in Java Agent 5.6
	`org.forgerock.openam.agents.config.allow.custom.login`
	Introduced in Java Agent 5.6
	Recognized from AM 7

Function	Custom login redirect, Default Login Redirect, Login redirect, Login Redirect (Default)
Type	Boolean: `true` returns true; all other strings return `false`.
Default	`false`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`

AM console	Tab: `AM Services (from AM 7)` Title: `Enable Custom Login Mode` Legacy title: `Allow Custom Login Mode`

AM Login URL List

The URL of the login page to use for authentication.

During the redirect, the agent appends the following parameters to the agent's CDSSO endpoint:

- The goto parameter configured in Goto Parameter Name
- A nonce parameter

Use the format `URL[?realm=realm_name?parameter1=value1&…]`, where:

- `URL`: URL of the login page to use for authentication
- `[?realm=realm_name&parameter1=value1&…]`: Optional parameters that the agent passes to the login page, for example, the AM realm at which to authenticate.

You do not need to specify an authentication realm if any of the following conditions are true:

- The custom login page sets the realm parameter, for example, because it lets the user choose the realm.
- The user authenticates into a realm that has DNS aliases configured in AM. AM then logs the user into the realm whose DNS alias matches the incoming request URL. For example, an inbound request from `http://marketplace.example.com` logs in the marketplace realm if the realm alias is set to `marketplace.example.com`.
- The user authenticates to the top-level realm.

This parameter can be overwritten by the custom login page if, for example, the user chooses the authentication realm.

Specify as many parameters your custom login pages require.

Example:

`https://login.example.com/login.jsp?realm=marketplace&param1=value1`

In some versions of AM you can configure more than one value for this property, but only the first value is honored.

Property name	`com.sun.identity.agents.config.login.url`
Aliases	`com.sun.identity.agents.config.login.url` Introduced in Java Agent 5.0 Recognized from AM 6
Function	Custom login redirect, Default Login Redirect, Login redirect, Login Redirect (Default)
Type	List

Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `AM Services` Title: `AM Login URL List` Legacy title: `AM Login URL`

Login Reason Parameter Name

When Enable Custom Login Mode is `true`, this property specifies the name of a parameter included in calls to the custom login URL, to indicate why authentication is required. The parameter value can be used in a custom login page to provide additional feedback to the authenticating user.

If this property is specified, the agent includes a parameter named with the property value, and including one of the following values:

- `NO_TOKEN`: No token present in the original request.
- `TOKEN_EXPIRED`: Expiry time of the JWT was in the past.
- `EXCEPTION`: An unknown exception occurred, either while parsing the JWT or at some other stage of authentication.

To reduce the risk of leaking useful information, use the property Login Reason Value Map to change the strings for the above values.

For example, specifying `org.forgerock.agents.login.reason.parameter.name=auth_reason` can cause the agent to redirect authentication to the following URL: `https://custom.example.com:8443/….login_endpoint?…&auth_reason=TOKEN_EXPIRED&…`

Do not enter a value that clashes with other parameters used for authentication; for example, `realm` or `goto`.

Property name	`org.forgerock.agents.login.reason.parameter.name`
Aliases	`org.forgerock.agents.login.reason.parameter.name` Introduced in Java Agent 5.7
Function	Custom login redirect, Login redirect
Type	String
Bootstrap property	No
Required property	No
Restart required	No

Local configuration file	`AgentConfig.properties`
AM console	Tab: `Miscellaneous (from AM 7)` Title: `Login Reason Parameter Name`

Legacy Login URL List

Adds parameters conditionally to legacy login URLs.

Format, with no spaces between values:

`domain/path|url?param1=value1&param2=value2`

Domain/path

The incoming request URL:

- Domain: For example, `example.com`. The agent must match the domain and its subdomains. For example, `example.com` matches `mydomain.example.com` and `www.example.com`. Domains can also include path information, for example, `example.com/market`, but cannot specify ports.
- Subdomain: For example, `mydomain.example.com`. The agent match the domain, the subdomain, and any sub-subdomain. For example, `mydomain.example.com` matches `true.mydomain.example.com`. Subdomains can include path information, for example, `mydomain.example.com/s6ecure`, but cannot specify ports.
- Path: For example, `/myapp`.
- No value: Nothing is specified before the `|` character and the rule applies to every incoming request.

URL

The URL to which redirect incoming login requests. The URL may be an AM instance, an AM site, or a 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/login.jsp`

`https://am.example.com:8443/openam/XUI/#login/`

`https://am.example.com:8443/openam/customlogin/login.jsp`

If the URL is not specified, the agent redirects the request to the AM instance or site specified by the following bootstrap properties:

`org.forgerock.agents.am.protocol://org.forgerock.agents.am.hostname:org.forgerock.agents.am.port/org.forgerock.agents.am.path`

&parameter1=value1

Parameters that can be added to the URL. Add as many parameters as your custom login pages need. Chain parameters with an & character, for example, `realm=value&parameter1=value1&parameter2=value2`.

Examples

`org.forgerock.agents.legacy.login.url.list[0]=example.com|https://am.example.com/openam/XUI/#login&realm=customers`

`org.forgerock.agents.legacy.login.url.list[1]=myapp.domain.com|https://login.example.com/apps/login.jsp?realm=sales`

`org.forgerock.agents.legacy.login.url.list[2]=sales.example.com/marketplace|?realm=marketplace`

`org.forgerock.agents.legacy.login.url.list[3]=|https://login.example.com/apps/login.jsp?realm=sales&isblue=true&carowner=true`

`org.forgerock.agents.legacy.login.url.list[4]=|?realm=sales`

Property name	`org.forgerock.agents.legacy.login.url.list`

Aliases	`org.forgerock.openam.agents.config.conditional.custom.login.url`
	Introduced in Java Agent 5.6
	Recognized from AM 7
	`org.forgerock.agents.legacy.login.url.list`
	Introduced in Java Agent 5.6

Function	Custom login redirect, Default Login Redirect, Login redirect, Login Redirect (Default)
Type	List
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `AM Services (from AM 7)` Title: `Legacy Login URL List` Legacy title: `Custom Conditional Login URL`

Default Login Redirect

OAuth Login URL List

Use only when Enable Custom Login Mode is `false` and AM Login URL List is empty.

Specify rules to evaluate the incoming request URL, based on domain, path, request header, or query parameters. Specify a URL for login redirect with optional parameters.

Format, with no spaces between values:

```
[domain/path][header:value][?param=value[,param=value]]|[URL][?param=value&param=value]
```

When an unauthenticated request URL matches a rule specified by this property, the agent redirects the request to the specified URL for login.

When this property configures multiple rules, the agent sorts the rules into the following order and applies them in that order until it finds a match:

1. Header specification - a rule with a header specification is applied before other rules
2. Longest domain
3. Longest path
4. Highest number of parameter specifications

During redirect, the agent appends the goto parameter configured in Goto Parameter Name, and a nonce parameter, to the agent's CDSSO endpoint. If Enable FQDN Checking is `true`, the agent iterates through the list of URLs until it finds a redirect URL that matches the FQDN check values. Otherwise, the agent redirects the user to the URL configured in the conditional redirect rules.

`[domain/path]`

The incoming request URL:

- Domain: For example, `example.com`. The agent must match the domain and its subdomains. For example, `example.com` matches `mydomain.example.com` and `www.example.com`. Domains can also include path information, for example, `example.com/market`, but cannot specify ports.
- Subdomain: For example, `mydomain.example.com`. The agent match the domain, the subdomain, and any sub-subdomain. For example, `mydomain.example.com` matches `true.mydomain.example.com`. Subdomains can include path information, for example, `mydomain.example.com/s6ecure`, but cannot specify ports.
- Path: For example, `/myapp`.
- No value: Nothing is specified before the | character and the rule applies to every incoming request.

`[header:value]`

One header/value pair provided in the incoming request. If the header value is not provided, the header can take any value. For example:

Requests containing a header called `X-local` with the value `provided` are redirected to the default URL:

```
org.forgerock.agents.oauth.login.url.list[0]= X-local:provided|
```

Requests containing a header called `X-local` with any value are redirected to the default URL:

```
org.forgerock.agents.oauth.login.url.list[0]= X-local:|
```

`[?param=value[,param=value]`

One or more parameter and value pairs provided in the incoming request. If the parameter value is not provided, the parameter can take any value. For example:

Requests containing a parameter called `site` with the value `shopping` are redirected to the default URL:

`org.forgerock.agents.oauth.login.url.list[2]= ?site=shopping|`

Requests containing a parameter called `target`with the value `cooking` **AND** a parameter called `price` with the value `low` are redirected to the default URL:+

`org.forgerock.openam.agents.config.conditional.login.url[0]= ?target=cooking,price=low|`

`[URL]`

The login URL. The URL can be an AM instance, an AM site, or a 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/login.jsp`

`https://am.example.com:8443/openam/XUI/#login/`

`https://am.example.com:8443/openam/customlogin/login.jsp`

If `[URL]` is not specified, the agent redirects the request to the AM instance or site specified by the following bootstrap properties:

`org.forgerock.agents.am.protocol://org.forgerock.agents.am.hostname:org.forgerock.agents.am.port/org.forgerock.agents.am.path`

`[?param=value&param=value]`

One or more parameters to add to the login URL. Chain multiple parameters with an ampersand (&), for example, `realm=value&parameter1=value1&parameter2=value2`.

When the parameter is `?realm=value` it specifies the AM realm into which the agent logs the users. For example, `?realm=marketplace`.

When redirecting to AM's XUI, use an ampersand (&) instead of a question mark (?). For example, `https://am.example.com:8443/openam/XUI/#login/&realm=marketplace`.

A realm parameter is not required in the login URL when any of the following conditions are true:

- The custom login page itself sets the realm parameter, for example, because it lets the user choose it. In this case, you must ensure the custom login page always returns a realm parameter to the agent.
- The realm that the agent is logging the user into has DNS aliases configured in AM.
- AM logs the user into the realm whose DNS alias matches the incoming request URL. For example, an inbound request from the http://marketplace.example.com URL logs in the marketplace realm if the realm alias is set to marketplace.example.com.
- The users should always log in to the Top Level Realm.

Examples

+

Requests containing a header called `X-local` with the value `provided` are redirected to the specified URL in the `beta` realm:

+

`org.forgerock.agents.oauth.login.url.list[0]= X-local:provided|http://mysite.local.com:8081/login?realm=beta`

+

Requests containing a header called `X-local` with any value are redirected to the default URL in the `gamma` realm:

+

`org.forgerock.agents.oauth.login.url.list[1]= X-local:|?realm=gamma`

+

Requests containing a parameter called `site` with the value `shopping` **AND** a parameter called `mode` with the value `discount` are redirected to the default URL in the `discountshopping` realm:

+

`org.forgerock.agents.oauth.login.url.list[2]= ?site=shopping,mode=discount|?realm=discountshopping`

+

Requests containing a parameter called `target` with the value `cooking` are redirected to the AM XUI page in the `kitchen` realm. Note the use of `&` before the realm parameter:

+

`org.forgerock.openam.agents.config.conditional.login.url[0]= ?target=cooking|https://am.example.com:8443/openam/XUI/#login/&realm=kitchen`

+

Requests containing a parameter called `target` with the value `cooking` are redirected to a non-AM login page in the `kitchen` realm. Note the use of `?` before the realm parameter:

+

`org.forgerock.openam.agents.config.conditional.login.url[0]= ?target=cooking|https://mysite.example.com:8443/login/?realm=kitchen`

Property name	`org.forgerock.agents.oauth.login.url.list`

Aliases	`org.forgerock.agents.oauth.login.url.list`
	Introduced in Java Agent 5.6
	`org.forgerock.openam.agents.config.conditional.login.url`
	Introduced in Java Agent 5.6
	Recognized from AM 6

Function	Custom login redirect, Default Login Redirect, Login redirect, Login Redirect (Default)
Type	List
Bootstrap property	No
Required property	No

Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `AM Services` Title: `OAuth Login URL List` Legacy title: `AM Conditional Login URL`

Enable Custom Login Mode

Set the login redirection mode, as follows:

- `false`: Use default login redirection mode.
  - The agent can redirect requests to any AM instance that supports the `/oauth2/authorize` endpoint. By default, this is the AM instance that is specified during installation.
  - The `/oauth2/authorize` endpoint returns an OIDC ID token. This is the only response the agent accepts.
  - Use with OAuth Login URL List to modify or redirect calls to the endpoint that provides the tokens.
- `true`: Use custom login redirection mode.
  - The agent handles JWTs or SSO tokens as session tokens for authentication and authorization, and can can redirect login anywhere.
  - Use with AM Login URL List and Legacy Login URL List to modify or redirect calls.

During session upgrade, the format of the composite advice is as follows:

- When both this property and Enable SSO Token Acceptance are `true`, the composite advice has the following format: `?authIndexType=composite_advice&authIndexValue=<Advices Value>`
- When either property is `false`, the composite advice has the following format: `?composite_advice=<Advices Value>`

Property name	`org.forgerock.agents.legacy.login.enabled`

Aliases	`org.forgerock.agents.legacy.login.enabled`
	Introduced in Java Agent 5.6
	`org.forgerock.openam.agents.config.allow.custom.login`
	Introduced in Java Agent 5.6
	Recognized from AM 7

Function	Custom login redirect, Default Login Redirect, Login redirect, Login Redirect (Default)
Type	Boolean: `true` returns true; all other strings return `false`.
Default	`false`

Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `AM Services (from AM 7)` Title: `Enable Custom Login Mode` Legacy title: `Allow Custom Login Mode`

AM Login URL List

The URL of the login page to use for authentication.

During the redirect, the agent appends the following parameters to the agent's CDSSO endpoint:

- The goto parameter configured in Goto Parameter Name
- A nonce parameter

Use the format `URL[?realm=realm_name?parameter1=value1&…]`, where:

- `URL`: URL of the login page to use for authentication
- `[?realm=realm_name&parameter1=value1&…]`: Optional parameters that the agent passes to the login page, for example, the AM realm at which to authenticate.

You do not need to specify an authentication realm if any of the following conditions are true:

- The custom login page sets the realm parameter, for example, because it lets the user choose the realm.
- The user authenticates into a realm that has DNS aliases configured in AM. AM then logs the user into the realm whose DNS alias matches the incoming request URL. For example, an inbound request from `http://marketplace.example.com` logs in the marketplace realm if the realm alias is set to `marketplace.example.com`.
- The user authenticates to the top-level realm.

This parameter can be overwritten by the custom login page if, for example, the user chooses the authentication realm.

Specify as many parameters your custom login pages require.

Example:

`https://login.example.com/login.jsp?realm=marketplace&param1=value1`

In some versions of AM you can configure more than one value for this property, but only the first value is honored.

Property name	`com.sun.identity.agents.config.login.url`

Aliases	`com.sun.identity.agents.config.login.url` Introduced in Java Agent 5.0 Recognized from AM 6
Function	Custom login redirect, Default Login Redirect, Login redirect, Login Redirect (Default)
Type	List
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `AM Services` Title: `AM Login URL List` Legacy title: `AM Login URL`

Legacy Login URL List

Adds parameters conditionally to legacy login URLs.

Format, with no spaces between values:

`domain/path|url?param1=value1&param2=value2`

Domain/path

The incoming request URL:

- Domain: For example, `example.com`. The agent must match the domain and its subdomains. For example, `example.com` matches `mydomain.example.com` and `www.example.com`. Domains can also include path information, for example, `example.com/market`, but cannot specify ports.
- Subdomain: For example, `mydomain.example.com`. The agent match the domain, the subdomain, and any sub-subdomain. For example, `mydomain.example.com` matches `true.mydomain.example.com`. Subdomains can include path information, for example, `mydomain.example.com/s6ecure`, but cannot specify ports.
- Path: For example, `/myapp`.
- No value: Nothing is specified before the `|` character and the rule applies to every incoming request.

URL

The URL to which redirect incoming login requests. The URL may be an AM instance, an AM site, or a 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/login.jsp`

`https://am.example.com:8443/openam/XUI/#login/`

`https://am.example.com:8443/openam/customlogin/login.jsp`

If the URL is not specified, the agent redirects the request to the AM instance or site specified by the following bootstrap properties:

`org.forgerock.agents.am.protocol://org.forgerock.agents.am.hostname:org.forgerock.agents.am.port/org.forgerock.agents.am.path`

&parameter1=value1

Parameters that can be added to the URL. Add as many parameters as your custom login pages need. Chain parameters with an & character, for example, `realm=value&parameter1=value1&parameter2=value2`.

Examples

`org.forgerock.agents.legacy.login.url.list[0]=example.com|https://am.example.com/openam/XUI/#login&realm=customers`

`org.forgerock.agents.legacy.login.url.list[1]=myapp.domain.com|https://login.example.com/apps/login.jsp?realm=sales`

`org.forgerock.agents.legacy.login.url.list[2]=sales.example.com/marketplace|?realm=marketplace`

`org.forgerock.agents.legacy.login.url.list[3]=|https://login.example.com/apps/login.jsp?realm=sales&isblue=true&carowner=true`

`org.forgerock.agents.legacy.login.url.list[4]=|?realm=sales`

Property name	`org.forgerock.agents.legacy.login.url.list`

Aliases	`org.forgerock.openam.agents.config.conditional.custom.login.url`
	Introduced in Java Agent 5.6
	Recognized from AM 7
	`org.forgerock.agents.legacy.login.url.list`
	Introduced in Java Agent 5.6

Function	Custom login redirect, Default Login Redirect, Login redirect, Login Redirect (Default)
Type	List
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`

AM console	Tab: `AM Services (from AM 7)` Title: `Legacy Login URL List` Legacy title: `Custom Conditional Login URL`

Deprecated

Login Attempt Limit (deprecated)

When the value of this property is greater than zero, it defines the maximum number of failed login attempts allowed during a browser session. After this number, the agent blocks requests from the user.

Specify a value greater than zero. For example, if the property is set to 3, the agent blocks the fourth login request.

Use this property to mitigate the risk of brute force attacks.

Property name	`org.forgerock.agents.login.attempt.limit.count`

Aliases	`org.forgerock.agents.login.attempt.limit.count`
	Introduced in Java Agent 5.6
	`com.sun.identity.agents.config.login.attempt.limit`
	Introduced in Java Agent 5.0
	Recognized from AM 6

Function	Deprecated
Type	Integer
Default	`0`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Global` Title: `Login Attempt Limit (deprecated)`

Login Attempt Limit Cookie Name (deprecated)

The name of the cookie used to record the number of login attempts.

Property name	`org.forgerock.agents.login.counter.cookie.name`

Aliases	`com.sun.identity.agents.config.login.counter.name`
	Introduced in Java Agent 5.0
	Recognized from AM 7
	`org.forgerock.agents.login.counter.cookie.name`
	Introduced in Java Agent 5.6

Function	Deprecated
Type	String
Default	`amFilterParam`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Global (from AM 7)` Title: `Login Attempt Limit Cookie Name (deprecated)`

Encryption

Encryption Class

The name of the class used by the agent to implement encryption and decryption. The class must implement both `com.iplanet.services.util.AMEncryption` and `com.iplanet.services.util.ConfigurableKey`.

> **Warning**
>
> After changing this property, you must re-encrypt the agent profile password. For more information, refer to Encrypted Agent Password.

Property name	`org.forgerock.agents.encryptor.classname`

Aliases	`org.forgerock.agents.encryptor.classname`
	Introduced in Java Agent 5.0
	`com.iplanet.security.encryptor`
	Introduced in Java Agent 5.0

Function	Encryption, Required
Type	Classname
Default	`org.forgerock.openam.shared.security.crypto.AESWrapEncryption`

Bootstrap property	Yes
Required property	Yes - If this property is missing, the agent fails to start
Restart required	No
Local configuration file	`AgentBootstrap.properties`

Encryption Key/Salt

The key/salt used by the default encryption class to perform encryption and decryption within the agent.

Generate a secure, random key with `agentadmin --key` or `agentadmin --rotate` commands as described in Rotate the agent profile password.

If this property is not set, the agent terminates with a configuration error.

> **Warning**
>
> If you are changing this property manually or with `agentadmin --key`, you must re-encrypt the agent profile password. For more information, refer to Encrypted Agent Password.

Property name	`org.forgerock.agents.encryption.key`

Aliases	`org.forgerock.agents.encryption.key`
	Introduced in Java Agent 5.10.1
	Recognized from AM 7.1
	`am.encryption.pwd`
	Introduced in Java Agent 5.0
	`com.sun.identity.client.encryptionKey`
	Introduced in Java Agent 5.0

Function	Authentication service, Encryption
Type	String
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`

Fragment

Fragment Relay URI

A URI to act as a dummy endpoint within the agent for capturing URL fragments in unauthenticated requests:

- When empty, unauthenticated requests to a URL with a fragment are authenticated and then redirected to the URL without the fragment.
- When set, unauthenticated requests are authenticated and then redirected to the requested URL. An extra redirect is incurred for all unauthenticated requests, to capture and process the URL fragment.

Use a dummy URI within the agent web application, such as `/agentapp/pre-authn-fragment-capture`. Avoid dummy URIs used for other purposes.

Property name	`org.forgerock.agents.authn.fragment.relay.uri`
Aliases	`org.forgerock.agents.authn.fragment.relay.uri` Introduced in Java Agent 5.7 Recognized from AM 7
Function	Fragment
Type	String
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Advanced (from AM 7)` Title: `Fragment Relay URI`

Fully qualified domain name

Default FQDN

The default FQDN to use when the agent cannot find a match in the FQDN Map. If this property is not defined, Enable FQDN Checking is set to `false`.

Use this property to map requests with virtual, invalid, or partial hostnames to URLs that contain a correct FQDN.

Property name	`org.forgerock.agents.fqdn.default`

Aliases	`org.forgerock.agents.fqdn.default`
	Introduced in Java Agent 5.6
	`com.sun.identity.agents.config.fqdn.default`
	Introduced in Java Agent 5.0
	Recognized from AM 6

Function	Fully qualified domain name
Type	String
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Global` Title: `Default FQDN` Legacy title: `FQDN Default`

FQDN Map

Key:Value maps of incoming hostname to outgoing domain. Use this property to map requests with virtual, invalid, or partial hostnames to a correct FQDN.

This property requires Enable FQDN Checking to be `true`, and Default FQDN to be set to suitable default FQDN.

The agent maintains the following maps, which can each contain multiple entries:

- Map 1, where the key is the incoming hostname without wildcards, and the value is the outgoing domain.
- Map 2, where the key is the incoming hostname with wildcards (* or ?), and the value is the outgoing domain.

Map keys are case insensitive. Incoming hostnames are converted to lowercase before the agent maps them, and the agent automatically converts uppercase keys and values to lowercase before mapping.

The agent maps FQDNs as follows:

1. Searches map 1 for the incoming hostname. If there is a match, the agent redirects the request to the mapped value.
2. Searches map 2 for a pattern that matches the incoming hostname, iterating through the entries in random order. If there is a match, the agent redirects the request to the mapped value.
3. Redirects the request to the hostname in Default FQDN.

Examples:

`org.forgerock.agents.fqdn.map[agent]=agent.localtest.me`

`org.forgerock.agents.fqdn.map[agent.virtualtest.me]=virtual-host.localtest.me`

`org.forgerock.agents.fqdn.map[agent-*.localtest.me]=agent.localtest.me`

Property name	`org.forgerock.agents.fqdn.map`

Aliases	`com.sun.identity.agents.config.fqdn.mapping`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.fqdn.map`
	Introduced in Java Agent 5.6

Function	Fully qualified domain name
Type	Map • Keys: canonical name of invalid server • Values: canonical name of valid server
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Global` Title: `FQDN Map` Legacy title: `FQDN Virtual Host Map`

Enable FQDN Checking

When `true`, the agent compares the hostname of each request to the mappings in FQDN Map.

- If it finds a match, it transforms the request URL to the mapped URL.
- If it doesn't find a match, it transforms the request URL to the value in Default FQDN.

If Default FQDN is not set, this property is automatically set to `false`.

Property name	`org.forgerock.agents.fqdn.check.enabled`

Aliases	`com.sun.identity.agents.config.fqdn.check.enable`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.fqdn.check.enabled`
	Introduced in Java Agent 5.6

Function	Fully qualified domain name

Type	Boolean: `true` returns true; all other strings return `false`.
Default	`true`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Global` Title: `Enable FQDN Checking` Legacy title: `FQDN Check`

Global

Enable Prometheus Monitoring

When `true`, the agent is monitored by Prometheus. When `false`, the agent is not monitored by Prometheus.

Property name	`org.forgerock.agents.prometheus.monitoring.enabled`
Aliases	`org.forgerock.agents.prometheus.monitoring.enabled` Introduced in Java Agent 5.6
Function	Global
Type	Boolean: `true` returns true; all other strings return `false`.
Default	`true`
Bootstrap property	Yes
Required property	No
Restart required	No
Local configuration file	`AgentBootstrap.properties`
AM console	Tab: `Monitoring` Title: `Enable Prometheus Monitoring`

HTTP 302 Redirect Data

When Enable HTTP 302 Redirects, this property specifies the data to return instead of an HTTP 302 Redirect.

Property name	`org.forgerock.agents.302.redirect.http.data`
Aliases	`org.forgerock.agents.302.redirect.http.data` Introduced in Java Agent 5.8.0 Recognized from AM 7.1
Function	Global
Type	String
Default	`{ "redirect": { "requestUri": "%REQUEST_URI%", "requestUrl": "%REQUEST_URL%", "targetUrl": "%TARGET%" } }`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Miscellaneous (from AM 7.1)` Title: `HTTP 302 Redirect Data`

HTTP 302 Redirect Not-Enforced List

When Enable HTTP 302 Redirects, this property specifies a list of URLs for which HTTP 302 Redirect does not take place.

If a request matches an entry in the list, HTTP 302 Redirect does not take place for that request, and the agent returns a block of configurable JSON.

Property name	`org.forgerock.agents.config.json.url`

Aliases	`org.forgerock.agents.config.json.url`
	Introduced in Java Agent 5.8.0
	`org.forgerock.agents.302.redirect.ner.list`
	Introduced in Java Agent 5.8.0
	Recognized from AM 7.1

Function	Global
Type	List

Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Miscellaneous (from AM 7.1)` Title: `HTTP 302 Redirect Not-Enforced List` Legacy title: `HTTP 302 Redirect Not Enforced List`

HTTP 302 Redirect Replacement HTTP Status Code

When Enable HTTP 302 Redirects is `false`, this property specifies the HTTP code to return instead of an HTTP 302 (Redirect).

Property name	`org.forgerock.agents.302.redirect.http.status.code`

Aliases	`org.forgerock.agents.302.redirect.http.status.code`
	Introduced in Java Agent 5.8.0
	Recognized from AM 7.1
	`org.forgerock.agents.config.json.response.code`
	Introduced in Java Agent 5.8.0

Function	Global
Type	Integer
Default	`200`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Miscellaneous (from AM 7.1)` Title: `HTTP 302 Redirect Replacement HTTP Status Code` Legacy title: `HTTP 302 Redirect Replacement HTTP Code`

Goto Parameter Name

Renames the `goto` parameter. During redirection, the agent appends the requested URL to the named parameter.

Use this property when your web application requires a parameter other than `goto`.

In the following example, the parameter is renamed to `goto2`:

```
com.sun.identity.agents.config.redirect.param=goto2
```

The redirection URL becomes like this:

```
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, configured with Access Denied URI Map.

Property name	`com.sun.identity.agents.config.redirect.param`
Aliases	`com.sun.identity.agents.config.redirect.param` Introduced in Java Agent 5.0 Recognized from AM 6
Function	Global
Type	String
Default	`goto`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Miscellaneous` Title: `Goto Parameter Name`

HTTP 302 Redirect Content Type

When Enable HTTP 302 Redirects, this property specifies the content type of the data to return instead of an HTTP 302 Redirect.

Property name	`org.forgerock.agents.302.redirect.http.content.type`
Aliases	`org.forgerock.agents.302.redirect.http.content.type` Introduced in Java Agent 5.8.0 Recognized from AM 7.1
Function	Global
Type	String

Default	`application/json`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Miscellaneous (from AM 7.1)` Title: `HTTP 302 Redirect Content Type`

HTTP 302 Redirect Invert Not-Enforced List

When Enable HTTP 302 Redirects is `false`, and this property is `true`, the agent inverts the meaning of HTTP 302 Redirect Not-Enforced List, so that it specifies a list of URLs for which HTTP 302 Redirect does take place.

Property name	`org.forgerock.agents.config.json.url.invert`

Aliases	`org.forgerock.agents.config.json.url.invert`
	Introduced in Java Agent 5.8.0
	`org.forgerock.agents.302.redirect.invert.enabled`
	Introduced in Java Agent 5.8.0
	Recognized from AM 7.1

Function	Global
Type	Boolean: `true` returns true; all other strings return `false`.
Default	`false`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Miscellaneous (from AM 7.1)` Title: `HTTP 302 Redirect Invert Not-Enforced List`

Enable HTTP 302 Redirects

Controls how the agent handles redirects, as follows:

- `true`: HTTP 302 Redirects are enabled. When an unauthenticated request is made, and not-enforced rules do not apply, the agent returns an HTTP 302 code to redirect the user to an authentication endpoint.
- `false`: HTTP 302 Redirects are disabled. When an unauthenticated request is made, the agent returns a block of configurable JSON that can be intercepted.

The returned HTTP code, content type, and data is configured by the following properties

- HTTP 302 Redirect Replacement HTTP Status Code
- HTTP 302 Redirect Content Type
- HTTP 302 Redirect Data

Lists of URLs in a `not-enforced` rule style, for which the data is produced are configured by the following properties

- HTTP 302 Redirect Not-Enforced List
- HTTP 302 Redirect Invert Not-Enforced List

Use this option when it is difficult to handle 302, for example, when the agent is accessed by a JavaScript application, or by something other than a browser.

Property name	`org.forgerock.agents.302.redirects.enabled`
Aliases	`org.forgerock.agents.302.redirects.enabled` Introduced in Java Agent 5.8.0 Recognized from AM 7.1
Function	Global
Type	Boolean: `true` returns true; all other strings return `false`.
Default	`true`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Miscellaneous (from AM 7.1)` Title: `Enable HTTP 302 Redirects` Legacy title: `HTTP 302 Redirects Enabled`

Locale

Locale Country

The agent country. Changing this has no effect.

Property name	`org.forgerock.agents.locale.country`

Aliases	`org.forgerock.agents.locale.country`
	Introduced in Java Agent 5.6
	`com.sun.identity.agents.config.locale.country`
	Introduced in Java Agent 5.0
	Recognized from AM 6

Function	Locale
Type	String
Default	`US`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Miscellaneous` Title: `Locale Country`

Locale Language

The agent language. Changing this has no effect.

Property name	`org.forgerock.agents.locale.language`

Aliases	`org.forgerock.agents.locale.language`
	Introduced in Java Agent 5.6
	`com.sun.identity.agents.config.locale.language`
	Introduced in Java Agent 5.0
	Recognized from AM 6

Function	Locale
Type	String

Default	`en`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Miscellaneous` Title: `Locale Language`

Login

Enable Redirect to AM Success URL

When `true`, the agent redirects to the success URL specified in the AM service, if any. If no success URL is specified in AM, the agent redirects to the original requested URL, if any.

When `false`, the agent redirects to the requested URL, if any.

Property name	`org.forgerock.agents.authn.success.redirect.session.url.enabled`
Aliases	`org.forgerock.agents.authn.success.redirect.session.url.enabled` Introduced in Java Agent 5.6.3.0 Recognized from AM 7
Function	Login
Type	Boolean: `true` returns true; all other strings return `false`.
Default	`false`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `AM Services (from AM 7)` Title: `Enable Redirect to AM Success URL` Legacy title: `Redirect to AM's Success URL`

Authentication Exchange Cookie Name

A cooke name that will be used by the authentication exchange endpoint. The value is empty by default, and the endpoint is not able to examine cookie values.

Property name	`org.forgerock.agents.authn.exchange.cookie.name`
Aliases	`org.forgerock.agents.authn.exchange.cookie.name` Introduced in Java Agent 5.7
Function	Login
Type	String
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `SSO (from AM 7)` Title: `Authentication Exchange Cookie Name`

Login Reason Value Map

When Login Reason Parameter Name is set, this property specifies alternative strings to use for the supported values. For example:

Consider the example where Login Reason Parameter Name is set to `auth_reason`, and this property is set as follows:

`org.forgerock.agents.login.reason.map[NO_TOKEN]=notoken`

`org.forgerock.agents.login.reason.map[TOKEN_EXPIRED]=expired`

`org.forgerock.agents.login.reason.map[EXCEPTION]=exception`

The agent redirects authentication to the following URL:

`https://custom.example.com:8443/….login_endpoint?…&auth_reason=notoken&…`

Property name	`org.forgerock.agents.login.reason.remapper`

Aliases	`org.forgerock.agents.login.reason.remapper`
	Introduced in Java Agent 5.7
	`org.forgerock.agents.login.reason.map`
	Introduced in Java Agent 5.7
	Recognized from AM 7

Function	Login
Type	Map • Keys: failed login reason code • Values: masked value
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Miscellaneous (from AM 7)` Title: `Login Reason Value Map`

Redirect Attempt Limit

When the value of this property is greater than zero, it defines the maximum number of redirects allowed for a browser session. After this number, the agent blocks the request.

Specify a value greater than zero. For example, if the property is set to 3, then the agent blocks the request on the fourth redirect.

Use this property to mitigate the risk of infinite redirection loops.

Property name	`org.forgerock.agents.redirect.attempt.limit`

Aliases	`org.forgerock.agents.redirect.attempt.limit`
	Introduced in Java Agent 5.6
	`com.sun.identity.agents.config.redirect.attempt.limit`
	Introduced in Java Agent 5.0
	Recognized from AM 6

Function	Login
Type	Integer
Default	`0`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`

AM console	Tab: `Global` Title: `Redirect Attempt Limit`

Authentication Exchange URI

This property allows the administrator to enable an endpoint to facilitate the exchange of SSO tokens for OIDC JWTs. The value is empty by default and thus the endpoint is not accessible.

Property name	`org.forgerock.agents.authn.exchange.uri`
Aliases	`org.forgerock.agents.authn.exchange.uri` Introduced in Java Agent 5.7 Recognized from AM 7
Function	Login
Type	String
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `SSO (from AM 7)` Title: `Authentication Exchange URI`

Login Redirect (Default)

OAuth Login URL List

Use only when Enable Custom Login Mode is `false` and AM Login URL List is empty.

Specify rules to evaluate the incoming request URL, based on domain, path, request header, or query parameters. Specify a URL for login redirect with optional parameters.

Format, with no spaces between values:

`[domain/path][header:value][?param=value[,param=value]]|[URL][?param=value&param=value]`

When an unauthenticated request URL matches a rule specified by this property, the agent redirects the request to the specified URL for login.

When this property configures multiple rules, the agent sorts the rules into the following order and applies them in that order until it finds a match:

1. Header specification - a rule with a header specification is applied before other rules
2. Longest domain
3. Longest path
4. Highest number of parameter specifications

During redirect, the agent appends the goto parameter configured in Goto Parameter Name, and a nonce parameter, to the agent's CDSSO endpoint. If Enable FQDN Checking is `true`, the agent iterates through the list of URLs until it finds a redirect URL that matches the FQDN check values. Otherwise, the agent redirects the user to the URL configured in the conditional redirect rules.

`[domain/path]`

The incoming request URL:

- Domain: For example, `example.com`. The agent must match the domain and its subdomains. For example, `example.com` matches `mydomain.example.com` and `www.example.com`. Domains can also include path information, for example, `example.com/market`, but cannot specify ports.
- Subdomain: For example, `mydomain.example.com`. The agent match the domain, the subdomain, and any sub-subdomain. For example, `mydomain.example.com` matches `true.mydomain.example.com`. Subdomains can include path information, for example, `mydomain.example.com/s6ecure`, but cannot specify ports.
- Path: For example, `/myapp`.
- No value: Nothing is specified before the | character and the rule applies to every incoming request.

`[header:value]`

One header/value pair provided in the incoming request. If the header value is not provided, the header can take any value. For example:

Requests containing a header called `X-local` with the value `provided` are redirected to the default URL:

```
org.forgerock.agents.oauth.login.url.list[0]= X-local:provided|
```

Requests containing a header called `X-local` with any value are redirected to the default URL:

```
org.forgerock.agents.oauth.login.url.list[0]= X-local:|
```

`[?param=value[,param=value]`

One or more parameter and value pairs provided in the incoming request. If the parameter value is not provided, the parameter can take any value. For example:

Requests containing a parameter called `site` with the value `shopping` are redirected to the default URL:

```
org.forgerock.agents.oauth.login.url.list[2]= ?site=shopping|
```

Requests containing a parameter called `target`with the value `cooking` **AND** a parameter called `price` with the value `low` are redirected to the default URL:+

```
org.forgerock.openam.agents.config.conditional.login.url[0]= ?target=cooking,price=low|
```

[URL]

The login URL. The URL can be an AM instance, an AM site, or a 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/login.jsp`

`https://am.example.com:8443/openam/XUI/#login/`

`https://am.example.com:8443/openam/customlogin/login.jsp`

If `[URL]` is not specified, the agent redirects the request to the AM instance or site specified by the following bootstrap properties:

`org.forgerock.agents.am.protocol://org.forgerock.agents.am.hostname:org.forgerock.agents.am.port/org.forgerock.agents.am.path`

[?param=value&param=value]

One or more parameters to add to the login URL. Chain multiple parameters with an ampersand (&), for example, `realm=value&parameter1=value1&parameter2=value2`.

When the parameter is `?realm=value` it specifies the AM realm into which the agent logs the users. For example, `?realm=marketplace`.

When redirecting to AM's XUI, use an ampersand (&) instead of a question mark (?). For example, `https://am.example.com:8443/openam/XUI/#login/&realm=marketplace`.

A realm parameter is not required in the login URL when any of the following conditions are true:

- The custom login page itself sets the realm parameter, for example, because it lets the user choose it. In this case, you must ensure the custom login page always returns a realm parameter to the agent.
- The realm that the agent is logging the user into has DNS aliases configured in AM.
- AM logs the user into the realm whose DNS alias matches the incoming request URL. For example, an inbound request from the http://marketplace.example.com URL logs in the marketplace realm if the realm alias is set to marketplace.example.com.
- The users should always log in to the Top Level Realm.

Examples

+

Requests containing a header called `X-local` with the value `provided` are redirected to the specified URL in the `beta` realm:

+

```
org.forgerock.agents.oauth.login.url.list[0]= X-local:provided|http://mysite.local.com:8081/login?realm=beta
```

+

Requests containing a header called `X-local` with any value are redirected to the default URL in the `gamma` realm:

+

`org.forgerock.agents.oauth.login.url.list[1]= X-local:|?realm=gamma`

+

Requests containing a parameter called `site` with the value `shopping` **AND** a parameter called `mode` with the value `discount` are redirected to the default URL in the `discountshopping` realm:

+

`org.forgerock.agents.oauth.login.url.list[2]= ?site=shopping,mode=discount|?realm=discountshopping`

+

Requests containing a parameter called `target` with the value `cooking` are redirected to the AM XUI page in the `kitchen` realm. Note the use of `&` before the realm parameter:

+

`org.forgerock.openam.agents.config.conditional.login.url[0]= ?target=cooking|https://am.example.com:8443/openam/XUI/#login/&realm=kitchen`

+

Requests containing a parameter called `target` with the value `cooking` are redirected to a non-AM login page in the `kitchen` realm. Note the use of `?` before the realm parameter:

+

`org.forgerock.openam.agents.config.conditional.login.url[0]= ?target=cooking|https://mysite.example.com:8443/login/?realm=kitchen`

Property name	`org.forgerock.agents.oauth.login.url.list`

Aliases	`org.forgerock.agents.oauth.login.url.list`
	Introduced in Java Agent 5.6
	`org.forgerock.openam.agents.config.conditional.login.url`
	Introduced in Java Agent 5.6
	Recognized from AM 6

Function	Custom login redirect, Default Login Redirect, Login redirect, Login Redirect (Default)
Type	List
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`

AM console	Tab: `AM Services` Title: `OAuth Login URL List` Legacy title: `AM Conditional Login URL`

Enable Custom Login Mode

Set the login redirection mode, as follows:

- `false`: Use default login redirection mode.
  - The agent can redirect requests to any AM instance that supports the `/oauth2/authorize` endpoint. By default, this is the AM instance that is specified during installation.
  - The `/oauth2/authorize` endpoint returns an OIDC ID token. This is the only response the agent accepts.
  - Use with OAuth Login URL List to modify or redirect calls to the endpoint that provides the tokens.
- `true`: Use custom login redirection mode.
  - The agent handles JWTs or SSO tokens as session tokens for authentication and authorization, and can can redirect login anywhere.
  - Use with AM Login URL List and Legacy Login URL List to modify or redirect calls.

During session upgrade, the format of the composite advice is as follows:

- When both this property and Enable SSO Token Acceptance are `true`, the composite advice has the following format: `?authIndexType=composite_advice&authIndexValue=<Advices Value>`
- When either property is `false`, the composite advice has the following format: `?composite_advice=<Advices Value>`

Property name	`org.forgerock.agents.legacy.login.enabled`

Aliases	`org.forgerock.agents.legacy.login.enabled`
	Introduced in Java Agent 5.6
	`org.forgerock.openam.agents.config.allow.custom.login`
	Introduced in Java Agent 5.6
	Recognized from AM 7

Function	Custom login redirect, Default Login Redirect, Login redirect, Login Redirect (Default)
Type	Boolean: `true` returns true; all other strings return `false`.
Default	`false`
Bootstrap property	No
Required property	No

Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `AM Services (from AM 7)` Title: `Enable Custom Login Mode` Legacy title: `Allow Custom Login Mode`

AM Login URL List

The URL of the login page to use for authentication.

During the redirect, the agent appends the following parameters to the agent's CDSSO endpoint:

- The goto parameter configured in Goto Parameter Name
- A nonce parameter

Use the format `URL[?realm=realm_name?parameter1=value1&…]`, where:

- `URL`: URL of the login page to use for authentication
- `[?realm=realm_name&parameter1=value1&…]`: Optional parameters that the agent passes to the login page, for example, the AM realm at which to authenticate.

You do not need to specify an authentication realm if any of the following conditions are true:

- The custom login page sets the realm parameter, for example, because it lets the user choose the realm.
- The user authenticates into a realm that has DNS aliases configured in AM. AM then logs the user into the realm whose DNS alias matches the incoming request URL. For example, an inbound request from `http://marketplace.example.com` logs in the marketplace realm if the realm alias is set to `marketplace.example.com`.
- The user authenticates to the top-level realm.

This parameter can be overwritten by the custom login page if, for example, the user chooses the authentication realm.

Specify as many parameters your custom login pages require.

Example:

`https://login.example.com/login.jsp?realm=marketplace&param1=value1`

In some versions of AM you can configure more than one value for this property, but only the first value is honored.

Property name	`com.sun.identity.agents.config.login.url`
Aliases	`com.sun.identity.agents.config.login.url` Introduced in Java Agent 5.0 Recognized from AM 6

Function	Custom login redirect, Default Login Redirect, Login redirect, Login Redirect (Default)
Type	List
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `AM Services` Title: `AM Login URL List` Legacy title: `AM Login URL`

Legacy Login URL List

Adds parameters conditionally to legacy login URLs.

Format, with no spaces between values:

`domain/path|url?param1=value1&param2=value2`

Domain/path

The incoming request URL:

- Domain: For example, `example.com`. The agent must match the domain and its subdomains. For example, `example.com` matches `mydomain.example.com` and `www.example.com`. Domains can also include path information, for example, `example.com/market`, but cannot specify ports.
- Subdomain: For example, `mydomain.example.com`. The agent match the domain, the subdomain, and any sub-subdomain. For example, `mydomain.example.com` matches `true.mydomain.example.com`. Subdomains can include path information, for example, `mydomain.example.com/s6ecure`, but cannot specify ports.
- Path: For example, `/myapp`.
- No value: Nothing is specified before the `|` character and the rule applies to every incoming request.

URL

The URL to which redirect incoming login requests. The URL may be an AM instance, an AM site, or a 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/login.jsp`

`https://am.example.com:8443/openam/XUI/#login/`

`https://am.example.com:8443/openam/customlogin/login.jsp`

If the URL is not specified, the agent redirects the request to the AM instance or site specified by the following bootstrap properties:

`org.forgerock.agents.am.protocol://org.forgerock.agents.am.hostname:org.forgerock.agents.am.port/org.forgerock.agents.am.path`

&parameter1=value1

Parameters that can be added to the URL. Add as many parameters as your custom login pages need. Chain parameters with an & character, for example, `realm=value&parameter1=value1&parameter2=value2`.

Examples

`org.forgerock.agents.legacy.login.url.list[0]=example.com|https://am.example.com/openam/XUI/#login&realm=customers`

`org.forgerock.agents.legacy.login.url.list[1]=myapp.domain.com|https://login.example.com/apps/login.jsp?realm=sales`

`org.forgerock.agents.legacy.login.url.list[2]=sales.example.com/marketplace|?realm=marketplace`

`org.forgerock.agents.legacy.login.url.list[3]=|https://login.example.com/apps/login.jsp?realm=sales&isblue=true&carowner=true`

`org.forgerock.agents.legacy.login.url.list[4]=|?realm=sales`

Property name	`org.forgerock.agents.legacy.login.url.list`

Aliases	`org.forgerock.openam.agents.config.conditional.custom.login.url`
	Introduced in Java Agent 5.6
	Recognized from AM 7
	`org.forgerock.agents.legacy.login.url.list`
	Introduced in Java Agent 5.6

Function	Custom login redirect, Default Login Redirect, Login redirect, Login Redirect (Default)
Type	List
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `AM Services (from AM 7)` Title: `Legacy Login URL List` Legacy title: `Custom Conditional Login URL`

Login redirect

Enable SSO Token Acceptance

Set this property as follows:

- `true`: Accept SSO tokens. Use this option when the agent and the token issuer are in the same domain, and for web applications and APIs where the backend accepts user information from SSO tokens.
- `false`: Do not accept SSO tokens; require OIDC JWTs for authentication.

During session upgrade the format of the composite advice is as follows:

- When both this property and Enable Custom Login Mode are `true`, the composite advice has the following format: `?authIndexType=composite_advice&authIndexValue=<Advices Value>`
- When either property is `false`, the composite advice has the following format: `?composite_advice=<Advices Value>`

Property name	`org.forgerock.agents.accept.sso.tokens.enabled`

Aliases	`org.forgerock.agents.accept.sso.tokens.enabled`
	Introduced in Java Agent 5.7.1
	Recognized from AM 7.1
	`org.forgerock.agents.accept.sso.tokens`
	Introduced in Java Agent 5.7.1
	`com.forgerock.agents.accept.sso.tokens`
	Introduced in Java Agent 5.7.1

Function	Custom login redirect, Login redirect, SSO cookie handling
Type	Boolean: `true` returns true; all other strings return `false`.
Default	`false`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `SSO (from AM 7.1)` Title: `Enable SSO Token Acceptance` Legacy title: `Accept SSO Tokens`

OAuth Login URL List

Use only when Enable Custom Login Mode is `false` and AM Login URL List is empty.

Specify rules to evaluate the incoming request URL, based on domain, path, request header, or query parameters. Specify a URL for login redirect with optional parameters.

Format, with no spaces between values:

```
[domain/path][header:value][?param=value[,param=value]]|[URL][?param=value&param=value]
```

When an unauthenticated request URL matches a rule specified by this property, the agent redirects the request to the specified URL for login.

When this property configures multiple rules, the agent sorts the rules into the following order and applies them in that order until it finds a match:

1. Header specification - a rule with a header specification is applied before other rules
2. Longest domain
3. Longest path
4. Highest number of parameter specifications

During redirect, the agent appends the goto parameter configured in Goto Parameter Name, and a nonce parameter, to the agent's CDSSO endpoint. If Enable FQDN Checking is `true`, the agent iterates through the list of URLs until it finds a redirect URL that matches the FQDN check values. Otherwise, the agent redirects the user to the URL configured in the conditional redirect rules.

`[domain/path]`

The incoming request URL:

- Domain: For example, `example.com`. The agent must match the domain and its subdomains. For example, `example.com` matches `mydomain.example.com` and `www.example.com`. Domains can also include path information, for example, `example.com/market`, but cannot specify ports.
- Subdomain: For example, `mydomain.example.com`. The agent match the domain, the subdomain, and any sub-subdomain. For example, `mydomain.example.com` matches `true.mydomain.example.com`. Subdomains can include path information, for example, `mydomain.example.com/s6ecure`, but cannot specify ports.
- Path: For example, `/myapp`.
- No value: Nothing is specified before the | character and the rule applies to every incoming request.

`[header:value]`

One header/value pair provided in the incoming request. If the header value is not provided, the header can take any value. For example:

Requests containing a header called `X-local` with the value `provided` are redirected to the default URL:

```
org.forgerock.agents.oauth.login.url.list[0]= X-local:provided|
```

Requests containing a header called `X-local` with any value are redirected to the default URL:

`org.forgerock.agents.oauth.login.url.list[0]= X-local:|`

[?param=value[,param=value]]

One or more parameter and value pairs provided in the incoming request. If the parameter value is not provided, the parameter can take any value. For example:

Requests containing a parameter called `site` with the value `shopping` are redirected to the default URL:

`org.forgerock.agents.oauth.login.url.list[2]= ?site=shopping|`

Requests containing a parameter called `target`with the value `cooking` **AND** a parameter called `price` with the value `low` are redirected to the default URL:+

`org.forgerock.openam.agents.config.conditional.login.url[0]= ?target=cooking,price=low|`

[URL]

The login URL. The URL can be an AM instance, an AM site, or a 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/login.jsp`

`https://am.example.com:8443/openam/XUI/#login/`

`https://am.example.com:8443/openam/customlogin/login.jsp`

If `[URL]` is not specified, the agent redirects the request to the AM instance or site specified by the following bootstrap properties:

`org.forgerock.agents.am.protocol://org.forgerock.agents.am.hostname:org.forgerock.agents.am.port/org.forgerock.agents.am.path`

[?param=value&param=value]

One or more parameters to add to the login URL. Chain multiple parameters with an ampersand (&), for example, `realm=value&parameter1=value1&parameter2=value2`.

When the parameter is `?realm=value` it specifies the AM realm into which the agent logs the users. For example, `?realm=marketplace`.

When redirecting to AM's XUI, use an ampersand (&) instead of a question mark (?). For example, `https://am.example.com:8443/openam/XUI/#login/&realm=marketplace`.

A realm parameter is not required in the login URL when any of the following conditions are true:

- The custom login page itself sets the realm parameter, for example, because it lets the user choose it. In this case, you must ensure the custom login page always returns a realm parameter to the agent.
- The realm that the agent is logging the user into has DNS aliases configured in AM.
- AM logs the user into the realm whose DNS alias matches the incoming request URL. For example, an inbound request from the http://marketplace.example.com URL logs in the marketplace realm if the realm alias is set to marketplace.example.com.
- The users should always log in to the Top Level Realm.

Examples

+

Requests containing a header called `X-local` with the value `provided` are redirected to the specified URL in the `beta` realm:

+

```
org.forgerock.agents.oauth.login.url.list[0]= X-local:provided|http://mysite.local.com:8081/login?realm=beta
```

+

Requests containing a header called `X-local` with any value are redirected to the default URL in the `gamma` realm:

+

```
org.forgerock.agents.oauth.login.url.list[1]= X-local:|?realm=gamma
```

+

Requests containing a parameter called `site` with the value `shopping` **AND** a parameter called `mode` with the value `discount` are redirected to the default URL in the `discountshopping` realm:

+

```
org.forgerock.agents.oauth.login.url.list[2]= ?site=shopping,mode=discount|?realm=discountshopping
```

+

Requests containing a parameter called `target` with the value `cooking` are redirected to the AM XUI page in the `kitchen` realm. Note the use of `&` before the realm parameter:

+

```
org.forgerock.openam.agents.config.conditional.login.url[0]= ?target=cooking|https://am.example.com:8443/openam/XUI/#login/&realm=kitchen
```

+

Requests containing a parameter called `target` with the value `cooking` are redirected to a non-AM login page in the `kitchen` realm. Note the use of `?` before the realm parameter:

+

```
org.forgerock.openam.agents.config.conditional.login.url[0]= ?target=cooking|https://mysite.example.com:8443/login/?realm=kitchen
```

Property name	`org.forgerock.agents.oauth.login.url.list`

Aliases	`org.forgerock.agents.oauth.login.url.list`
	Introduced in Java Agent 5.6
	`org.forgerock.openam.agents.config.conditional.login.url`
	Introduced in Java Agent 5.6
	Recognized from AM 6

Function	Custom login redirect, Default Login Redirect, Login redirect, Login Redirect (Default)
Type	List
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `AM Services` Title: `OAuth Login URL List` Legacy title: `AM Conditional Login URL`

Enable Custom Login Mode

Set the login redirection mode, as follows:

- `false`: Use default login redirection mode.
  - The agent can redirect requests to any AM instance that supports the `/oauth2/authorize` endpoint. By default, this is the AM instance that is specified during installation.
  - The `/oauth2/authorize` endpoint returns an OIDC ID token. This is the only response the agent accepts.
  - Use with OAuth Login URL List to modify or redirect calls to the endpoint that provides the tokens.
- `true`: Use custom login redirection mode.
  - The agent handles JWTs or SSO tokens as session tokens for authentication and authorization, and can can redirect login anywhere.
  - Use with AM Login URL List and Legacy Login URL List to modify or redirect calls.

During session upgrade, the format of the composite advice is as follows:

- When both this property and Enable SSO Token Acceptance are `true`, the composite advice has the following format: `?authIndexType=composite_advice&authIndexValue=<Advices Value>`
- When either property is `false`, the composite advice has the following format: `?composite_advice=<Advices Value>`

Property name	`org.forgerock.agents.legacy.login.enabled`

Aliases	`org.forgerock.agents.legacy.login.enabled`
	Introduced in Java Agent 5.6
	`org.forgerock.openam.agents.config.allow.custom.login`
	Introduced in Java Agent 5.6
	Recognized from AM 7

Function	Custom login redirect, Default Login Redirect, Login redirect, Login Redirect (Default)
Type	Boolean: `true` returns true; all other strings return `false`.
Default	`false`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `AM Services (from AM 7)` Title: `Enable Custom Login Mode` Legacy title: `Allow Custom Login Mode`

AM Login URL List

The URL of the login page to use for authentication.

During the redirect, the agent appends the following parameters to the agent's CDSSO endpoint:

- The goto parameter configured in Goto Parameter Name
- A nonce parameter

Use the format `URL[?realm=realm_name?parameter1=value1&…]`, where:

- `URL`: URL of the login page to use for authentication
- `[?realm=realm_name&parameter1=value1&…]`: Optional parameters that the agent passes to the login page, for example, the AM realm at which to authenticate.

You do not need to specify an authentication realm if any of the following conditions are true:

- The custom login page sets the realm parameter, for example, because it lets the user choose the realm.
- The user authenticates into a realm that has DNS aliases configured in AM. AM then logs the user into the realm whose DNS alias matches the incoming request URL. For example, an inbound request from `http://marketplace.example.com` logs in the marketplace realm if the realm alias is set to `marketplace.example.com`.
- The user authenticates to the top-level realm.

This parameter can be overwritten by the custom login page if, for example, the user chooses the authentication realm.

Specify as many parameters your custom login pages require.

Example:

```
https://login.example.com/login.jsp?realm=marketplace&param1=value1
```

In some versions of AM you can configure more than one value for this property, but only the first value is honored.

Property name	`com.sun.identity.agents.config.login.url`
Aliases	`com.sun.identity.agents.config.login.url` Introduced in Java Agent 5.0 Recognized from AM 6
Function	Custom login redirect, Default Login Redirect, Login redirect, Login Redirect (Default)
Type	List
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `AM Services` Title: `AM Login URL List` Legacy title: `AM Login URL`

Login Reason Parameter Name

When Enable Custom Login Mode is `true`, this property specifies the name of a parameter included in calls to the custom login URL, to indicate why authentication is required. The parameter value can be used in a custom login page to provide additional feedback to the authenticating user.

If this property is specified, the agent includes a parameter named with the property value, and including one of the following values:

- `NO_TOKEN`: No token present in the original request.
- `TOKEN_EXPIRED`: Expiry time of the JWT was in the past.
- `EXCEPTION`: An unknown exception occurred, either while parsing the JWT or at some other stage of authentication.

To reduce the risk of leaking useful information, use the property Login Reason Value Map to change the strings for the above values.

For example, specifying `org.forgerock.agents.login.reason.parameter.name=auth_reason` can cause the agent to redirect authentication to the following URL: `https://custom.example.com:8443/….login_endpoint?…&auth_reason=TOKEN_EXPIRED&…`

Do not enter a value that clashes with other parameters used for authentication; for example, `realm` or `goto`.

Property name	`org.forgerock.agents.login.reason.parameter.name`
Aliases	`org.forgerock.agents.login.reason.parameter.name` Introduced in Java Agent 5.7
Function	Custom login redirect, Login redirect
Type	String
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Miscellaneous (from AM 7)` Title: `Login Reason Parameter Name`

Legacy Login URL List

Adds parameters conditionally to legacy login URLs.

Format, with no spaces between values:

`domain/path|url?param1=value1&param2=value2`

Domain/path

The incoming request URL:

- Domain: For example, `example.com`. The agent must match the domain and its subdomains. For example, `example.com` matches `mydomain.example.com` and `www.example.com`. Domains can also include path information, for example, `example.com/market`, but cannot specify ports.
- Subdomain: For example, `mydomain.example.com`. The agent match the domain, the subdomain, and any sub-subdomain. For example, `mydomain.example.com` matches `true.mydomain.example.com`. Subdomains can include path information, for example, `mydomain.example.com/s6ecure`, but cannot specify ports.
- Path: For example, `/myapp`.
- No value: Nothing is specified before the `|` character and the rule applies to every incoming request.

URL

The URL to which redirect incoming login requests. The URL may be an AM instance, an AM site, or a 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/login.jsp`

`https://am.example.com:8443/openam/XUI/#login/`

`https://am.example.com:8443/openam/customlogin/login.jsp`

If the URL is not specified, the agent redirects the request to the AM instance or site specified by the following bootstrap properties:

`org.forgerock.agents.am.protocol://org.forgerock.agents.am.hostname:org.forgerock.agents.am.port/org.forgerock.agents.am.path`

&parameter1=value1

Parameters that can be added to the URL. Add as many parameters as your custom login pages need. Chain parameters with an & character, for example, `realm=value&parameter1=value1&parameter2=value2`.

Examples

`org.forgerock.agents.legacy.login.url.list[0]=example.com|https://am.example.com/openam/XUI/#login&realm=customers`

`org.forgerock.agents.legacy.login.url.list[1]=myapp.domain.com|https://login.example.com/apps/login.jsp?realm=sales`

`org.forgerock.agents.legacy.login.url.list[2]=sales.example.com/marketplace|?realm=marketplace`

`org.forgerock.agents.legacy.login.url.list[3]=|https://login.example.com/apps/login.jsp?realm=sales&isblue=true&carowner=true`

`org.forgerock.agents.legacy.login.url.list[4]=|?realm=sales`

Property name	`org.forgerock.agents.legacy.login.url.list`

Aliases	`org.forgerock.openam.agents.config.conditional.custom.login.url`
	Introduced in Java Agent 5.6
	Recognized from AM 7
	`org.forgerock.agents.legacy.login.url.list`
	Introduced in Java Agent 5.6

Function	Custom login redirect, Default Login Redirect, Login redirect, Login Redirect (Default)
Type	List
Bootstrap property	No

Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `AM Services (from AM 7)` Title: `Legacy Login URL List` Legacy title: `Custom Conditional Login URL`

Logout

Logout URI Map

A map of request URIs that trigger logout of the user session. Use the following key:value format:

`web application name:logout URI`

When a URL that includes the specified URI is invoked, the agent kills the current session. It invokes the AM REST logout endpoint or the endpoint configured by Conditional Logout URL List.

The agent must be able to access the context for the URL. When a web application is specified, it must exist and the agent must have access to it.

The URL is a dummy URL. Even if a resource exists at the URL, it is never accessed.

Although logout can be triggered within any web application by invoking a single, common, URI, you can taylor it for particular web applications:

- To specify a single, common, URI, leave the key empty.
- To specify a URI unique to a particular web application, specify the name of the web application as the key.

Property name	`org.forgerock.agents.logout.endpoint.map`

Aliases	`com.sun.identity.agents.config.logout.uri`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.logout.endpoint.map`
	Introduced in Java Agent 5.6

Function	Logout
Type	Map • Keys: web application • Values: URI of dummy endpoint which will trigger logout
Bootstrap property	No

Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Application` Title: `Logout URI Map` Legacy title: `Application Logout URI`

Logout Request Parameter Map

A map of request parameters that trigger logout of the user session. Use the following key:value format:

```
Logout Request Parameter Map[web application name] = parameter name to trigger logout
```

The agent searches every incoming request for the parameter. When the agent detects the parameter, it invokes AM to kill the current session for the specified web application or all web applications.

The request URL must contain the parameter but does not need to assign a value to the parameter.

Although logout can be triggered from any web application by invoking one common URI, you can adapt it for particular web applications:

- To specify one parameter for all web applications, leave the key empty.
- To specify a parameter for a particular web application, specify the name of the web application as the key.

Property name	`org.forgerock.agents.logout.request.param.map`

Aliases	`org.forgerock.agents.logout.request.param.map`
	Introduced in Java Agent 5.6
	`com.sun.identity.agents.config.logout.request.param`
	Introduced in Java Agent 5.0
	Recognized from AM 6

Function	Logout
Type	Map • Keys: web application • Values: (single) HTTP query parameter to trigger logout
Bootstrap property	No
Required property	No
Restart required	No

Local configuration file	`AgentConfig.properties`
AM console	Tab: `Application` Title: `Logout Request Parameter Map` Legacy title: `Logout Request Parameter`

Always invalidate sessions

When `false`, the agent does not invoke the AM REST logout endpoint to kill the user session.

If Conditional Logout URL List is configured with a URL that does not perform a REST logout to AM, set this property to `true`. The agent additionally invokes the AM REST logout endpoint to invalidate the session.

Property name	`org.forgerock.agents.config.logout.session.invalidate.enabled`

Aliases	`org.forgerock.agents.config.logout.session.invalidate.enabled`
	Introduced in Java Agent 5.10.1
	`org.forgerock.agents.config.logout.session.invalidate`
	Introduced in Java Agent 5.10.1

Function	Logout
Type	Boolean: `true` returns true; all other strings return `false`.
Default	`true`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`

Enable Logout Introspection

When `true`, the agent checks the HTTP request body to locate the value of Logout Request Parameter Map.

Property name	`org.forgerock.agents.logout.introspection.enabled`

Aliases	`com.sun.identity.agents.config.logout.introspect.enabled`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.logout.introspection.enabled`
	Introduced in Java Agent 5.6

Function	Logout
Type	Boolean: `true` returns true; all other strings return `false`.
Default	`false`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Application` Title: `Enable Logout Introspection` Legacy title: `Logout Introspect Enabled`

Conditional Logout URL List

Define URLs to which the agent can conditionally direct the user on logout. This property does not trigger logout.

Configure one or more conditions, using this format:

`domain/path|url?param1=value1&param2=value2`

The request URL is compared to each condition in the list until a match is found. Conditions are evaluated by order of length, starting with the longest, irrespective of their order in the list.

Depending on the value of the redirection URL, perform this additional configuration:

- If the URL doesn't perform a REST logout to AM, set Always invalidate sessions to `true`. The agent additionally invokes the AM REST logout endpoint to invalidate the session.
- If the URL isn't relative to an AM URL, or in the same scheme, FQDN, and port as an AM URL, add it to the AM validation service.

In the following example, `example.com/path` is evaluated before `example.com`; the default condition is the shortest, and is evaluated last:

`org.forgerock.agents.conditional.logout.url.list[0]=example.com|?additional=value`

`org.forgerock.agents.conditional.logout.url.list[1]=example.com/path|?one=red&two=green&three=blue`

`org.forgerock.agents.conditional.logout.url.list[2]=mybank.com|http://mybank.com/myapp/logout?param=override`

`org.forgerock.agents.conditional.logout.url.list[3]=|?alpha=beta`

For more information, refer to Conditionally log out to different URLs.

Property name	`org.forgerock.agents.conditional.logout.url.list`

Aliases	`org.forgerock.openam.agents.config.conditional.logout.url`
	Introduced in Java Agent 5.6
	Recognized from AM 6
	`org.forgerock.agents.conditional.logout.url.list`
	Introduced in Java Agent 5.6

Function	Logout
Type	List
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `AM Services` Title: `Conditional Logout URL List` Legacy title: `AM Conditional Logout URL`

Logout Entry URI Map

A map of redirection resources to which the agent redirects the user after logout triggered by Logout URI Map or Logout Request Parameter Map. The resource can be an HTML page or JSP file. Use the following key:value format:

```
Logout Entry URI Map[web application name] = logout resource
```

To configure logout redirection for a specific web application, set the key to the name of the web application. To configure logout redirection for all web applications, leave the key empty (`Logout Entry URI Map = default logout resource`).

Specified resources are automatically added to the not-enforced list so that they can be accessed without authentication.

Depending on the type and value of the redirection resource, perform this additional configuration:

- If the resource is a URL that doesn't perform a REST logout to AM, set Always invalidate sessions to `true`. The agent additionally invokes the AM REST logout endpoint to invalidate the session.
- If the resource is a URL that isn't relative to an AM URL, or in the same scheme, FQDN, and port as an AM URL, add it to the AM validation service.

For more information, refer to Conditionally log out to different URLs.

Property name	`org.forgerock.agents.logout.goto.map`

Aliases	`com.sun.identity.agents.config.logout.entry.uri`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.logout.goto.map`
	Introduced in Java Agent 5.6

Function	Logout
Type	Map • Keys: web application • Values: URI of page explaining the user has been logged out
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Application` Title: `Logout Entry URI Map` Legacy title: `Logout Entry URI`

Logs

Agent Debug Level

The agent log level.

Make sure your container captures messages written to the standard output. Some containers do not, and warnings or critical errors can disappear forever.

Property name	`org.forgerock.agents.debug.level`

Aliases	`com.iplanet.services.debug.level`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.debug.level`
	Introduced in Java Agent 5.6

Function	Logs

Supported settings	***OFF*** (Deprecated) Uses the same log level as `ERROR`. To disable logging, edit `agent-logback.xml`. Forgerock advises not to disable logging. ***ERROR*** Log only errors. ***WARNING*** Log errors and warnings. ***MESSAGE*** Log errors, warnings, and informative messages. ***DEBUG*** Log errors, warnings, informative messages, and some debugging messages. ***TRACE*** Log fine-grained information, when you need the full visibility of what is happening in your application. This log level can create big log files, so use only when necessary, and then reduce the log level. ***ON*** (Deprecated) Uses the same log level as `TRACE`. When the log level is `ON`, `TRACE` level logs are written to file. In previous releases, `TRACE` level logs were written to the standard output. Make sure your container captures messages written to the standard output.
Default	`ERROR`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Global` Title: `Agent Debug Level`

Miscellaneous

Redirect Attempt Cookie Name

The cookie name to use to detect redirect loops while authenticating, which would indicate a cookie domain problem.

Property name	`org.forgerock.agents.redirect.cookie.name`

Aliases	`org.forgerock.agents.redirect.cookie.name`
	Introduced in Java Agent 5.6
	`com.sun.identity.agents.config.redirect.cookie.name`
	Introduced in Java Agent 5.0
	Recognized from AM 7

Function	Miscellaneous
Type	String
Default	`amFilterRDParam`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Global (from AM 7)` Title: `Redirect Attempt Cookie Name`

Encrypted Agent Password

The agent profile password, which must correspond to the value in AM.

Set this property to the encrypted value of the password, where the password is encrypted using the key in the property Encryption Key/Salt.

Use the following command to get the encrypted value of the password, where `passwordFile` contains only the password followed by a newline, and has the access permission `400`:

```
$ ./agentadmin --encrypt agentInstance passwordFile
```

Default: Empty

Property name	`org.forgerock.agents.encrypted.password`

Aliases	`com.iplanet.am.service.secret`
	Introduced in Java Agent 5.0
	`org.forgerock.agents.encrypted.password`
	Introduced in Java Agent 5.6

Function	Miscellaneous, Required
Type	String
Bootstrap property	No

Required property	Yes - If this property is missing, the agent fails to start
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentPassword.properties`

Enable Ignore Path Info

When `true`, when the request URL contains a wildcard '*' character, the path info and query are stripped from the URL before it is compared with the list of not-enforced URLs.

Property name	`org.forgerock.agents.ignore.path.info.enabled`

Aliases	`com.sun.identity.agents.config.ignore.path.info`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.ignore.path.info.enabled`
	Introduced in Java Agent 5.6

Function	Miscellaneous
Type	Boolean: `true` returns true; all other strings return `false`.
Default	`false`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Miscellaneous` Title: `Enable Ignore Path Info` Legacy title: `Ignore Path Info for Not Enforced URLs`

Custom Response Header Map

A key:value map of custom headers set by the agent for the client, where the key is the header name, and the value is the header value. For example, `org.forgerock.agents.response.header.map[Cache-Control]=no-cache`

Format `org.forgerock.agents.response.header.map[HEADER-NAME]=HEADER-VALUE`

Property name	`org.forgerock.agents.response.header.map`

Aliases	`org.forgerock.agents.response.header.map`
	Introduced in Java Agent 5.6
	`com.sun.identity.agents.config.response.header`
	Introduced in Java Agent 5.0
	Recognized from AM 6

Function	Miscellaneous
Type	Map • Keys: from header • Values: to header
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Global` Title: `Custom Response Header Map` Legacy title: `Custom Response Header`

Idle Time Refresh Window

The interval in minutes at which the agent calls AM to refresh a session idle timeout.

AM sessions have an idle timeout after which they expire. In general, when users access protected resources through an agent, the agent requests a policy decision on behalf of that user, which resets the idle timeout.

When the agent does not need to contact AM frequently, for example, when policy evaluation is already cached, sessions may unexpectedly expire in AM before the user has finished accessing the application.

Agents make one call per active user session at the end of the time interval, provided that the user is actively accessing the web application or site. If the user does not access the application during the configured window interval time, the agent will not make the call to AM at the end of the interval. Eventually, if the user is inactive for enough time, AM will log them out when the session reaches its idle timeout.

Configuring the idle timeout window to a short value, such as one minute, achieves a good balance between making additional calls to AM and providing a good user experience.

Increase this value only if the performance impact of making an extra call to AM every minute is noticeable enough in your environment.

Property name	`org.forgerock.agents.idle.time.window.minutes`

Aliases	`org.forgerock.agents.idle.time.window.minutes` Introduced in Java Agent 5.6.2.1 Recognized from AM 7
Function	Miscellaneous
Type	Integer
Default	`1`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Advanced (from AM 7)` Title: `Idle Time Refresh Window`

Service Resolver Class Name

The Java class name of the service resolver used to override the ForgeRock provided service resolver. Use this property to customize pre-handlers and post-handlers.

Property name	`org.forgerock.agents.service.resolver.class.name`
Aliases	`org.forgerock.agents.service.resolver.class.name` Introduced in Java Agent 5.6.2.1 Recognized from AM 7
Function	Miscellaneous
Type	Classname
Default	`com.sun.identity.agents.arch.ServiceResolver`
Bootstrap property	Yes
Required property	No
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentBootstrap.properties`

HTTP Session Binding

When `true`, the agent invalidates the HTTP session in these circumstances:

- Login failure
- When the user has no SSO session
- When the principal user name does not match the SSO user name

Property name	`org.forgerock.agents.http.session.binding.enabled`

Aliases	`com.sun.identity.agents.config.httpsession.binding`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.http.session.binding.enabled`
	Introduced in Java Agent 5.6

Function	Miscellaneous
Type	Boolean: `true` returns true; all other strings return `false`.
Default	`true`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Global` Title: `HTTP Session Binding`

Public AM URL

The assembled "public" URL of AM. This URL is used by the agent to redirect the user's browser to AM for login (customised or not), or if necessary, exchange an SSO token for a JWT.

The following properties make up the URL:

- AM Authentication Service Protocol
- AM Authentication Service Host Name
- AM Authentication Service Port
- AM Authentication Service Path

The "private" URL is used by the agent for tasks such as establishing websockets, and obtaining authentication tokens or session information. The AM or load balancer instance can be behind a firewall to which the agent has access.

Define this property when public access to AM is restricted to a different URL from the private URL.

Property name	`org.forgerock.agents.public.am.url`

Aliases	`com.forgerock.agents.public.am.url`
	Introduced in Java Agent 5.6.3.0
	`org.forgerock.agents.public.am.url`
	Introduced in Java Agent 5.6.3.0

Function	Miscellaneous, Required
Type	String
Bootstrap property	Yes
Required property	Yes - If this property is missing, the agent fails to start
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentBootstrap.properties`

Monitoring

Export Monitoring Metrics to CSV

When `true`, enables the export of agent performance monitoring metrics to comma-separated value (CSV) files.

Files are written the same directory as the agent instance debug files, for example in `/path/to/java_agents/tomcat_agent/Agent_001/logs/debug/`.

Property name	`org.forgerock.agents.config.monitoring.to.csv`

Aliases	`org.forgerock.agents.config.monitoring.to.csv`
	Introduced in Java Agent 5.6
	Recognized from AM 7
	`org.forgerock.agents.monitoring.to.csv.enabled`
	Introduced in Java Agent 5.6

Function	Monitoring
Type	Boolean: `true` returns true; all other strings return `false`.
Default	`false`
Bootstrap property	No

Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Advanced (from AM 7)` Title: `Export Monitoring Metrics to CSV`

CSV Monitoring Directory

The full path to the directory where the agent writes CSV monitoring files, when CSV monitoring is enabled.

The default is set by the installer and written to the bootstrap properties file.

Default: `/logs/debug` directory relative to the definedBy of the agent installation

Property name	`org.forgerock.agents.csv.monitoring.directory`
Aliases	`org.forgerock.agents.csv.monitoring.directory` Introduced in Java Agent 5.7
Function	Monitoring
Type	String
Bootstrap property	Yes
Required property	No
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentBootstrap.properties`

Not-enforced

Not-Enforced URIs

A space-delimited list of URIs. The agent applies the not-enforced rule for requests to the listed URIs. In the following example, the agent allows requests to the public and images directories of `myapp.example.com`:

`org.forgerock.agents.notenforced.uri.list=https://myapp.example.com:8443/public/* https://myapp.example.com:8443/images/*`

For more information and examples, see Conventions for not-enforced rules.

Property name	`org.forgerock.agents.notenforced.uri.list`

Aliases	`org.forgerock.agents.notenforced.uri.list`
	Introduced in Java Agent 5.6
	`com.sun.identity.agents.config.notenforced.uri`
	Introduced in Java Agent 5.0
	Recognized from AM 6

Function	Not-enforced
Type	List
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`

Not-Enforced Client IP List

A space-delimited list of IP addresses or network CIDR notation addresses. The agent applies the not-enforced rule to requests from the listed IP addresses.

Supported values are IPV4 and IPV6 addresses, IPV4 and IPV6 ranges of addresses delimited by the - character, and network ranges specified in CIDR notation.

Property name	`org.forgerock.agents.notenforced.ip.list`

Aliases	`org.forgerock.agents.notenforced.ip.list`
	Introduced in Java Agent 5.6
	`com.sun.identity.agents.config.notenforced.ip`
	Introduced in Java Agent 5.0
	Recognized from AM 6

Function	Not-enforced
Type	List
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`

Not-Enforced Favicon

When `true`, the agent does not enforce access to any files named `favicon.ico`, by inserting an internal not-enforced rule of `GET */favicon.ico`.

Property name	`org.forgerock.agents.auto.not.enforce.favicon`

Aliases	`org.forgerock.agents.auto.not.enforce.favicon`
	Introduced in Java Agent 5.7
	Recognized from AM 7
	`org.forgerock.agents.auto.not.enforce.favicon.enabled`
	Introduced in Java Agent 5.7

Function	Not-enforced
Type	Boolean: `true` returns true; all other strings return `false`.
Default	`true`
Bootstrap property	Yes
Required property	No
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentBootstrap.properties`
AM console	Tab: `Application (from AM 7)` Title: `Not-Enforced Favicon` Legacy title: `Not Enforced Favicon`

Enable Not-Enforced URIs Cache

When `true`, the agent caches evaluations of the Not-Enforced URIs.

Enable this setting when configuring many rules.

Property name	`org.forgerock.agents.notenforced.uri.cache.enabled`

Aliases	`org.forgerock.agents.notenforced.uri.cache.enabled`
	Introduced in Java Agent 5.6
	`com.sun.identity.agents.config.notenforced.uri.cache.enable`
	Introduced in Java Agent 5.0
	Recognized from AM 6

Function	Not-enforced

Type	Boolean: `true` returns true; all other strings return `false`.
Default	`true`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Application` Title: `Enable Not-Enforced URIs Cache` Legacy title: `Not Enforced URIs Cache Enabled`

Container Character Encoding

The character encoding used by the agent when encoding extended characters in the resource paths of not-enforced rules.

Property name	`org.forgerock.agents.container.encoding`
Aliases	`org.forgerock.agents.container.encoding` Introduced in Java Agent 5.9.1
Function	Container, Not-enforced
Type	String
Default	`UTF-8`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`

Java Class for Matching Not Enforced Rules

The Java class used to match URIs and IP addresses embedded within not enforced rules.

The specified class must implement the interface `com.sun.identity.agents.common.RulePatternMatcher`.

If the class fails to instantiate, an error is logged and the default NotEnforcedRulePatternMatcher is created instead.

Property name	`org.forgerock.agents.not.enforced.rule.pattern.matcher.classname`
Aliases	`org.forgerock.agents.not.enforced.rule.pattern.matcher.classname` Introduced in Java Agent 5.9.0
Function	Not-enforced
Type	Classname
Default	`com.sun.identity.agents.common.NotEnforcedRulePatternMatcher`
Bootstrap property	Yes
Required property	No
Restart required	No
Local configuration file	`AgentBootstrap.properties`

Not-Enforced Compound Rule Separator

A delimiter for not-enforced compound rules. The delimiter can be a single character or a string. For example, setting the delimiter to `&&` allows compound rules to be specified as:

```
GET 10.5.1.5 100.2.21.36 && /public/*
```

```
REGEX 10\.4\.3\.5 && [^/]+\/free.jpg
```

Property name	`org.forgerock.agents.notenforced.compound.separator`

Aliases	`org.forgerock.agents.notenforced.compound.separator`
	Introduced in Java Agent 5.6
	`com.sun.identity.agents.config.notenforced.rule.compound.separator`
	Introduced in Java Agent 5.0
	Recognized from AM 7

Function	Not-enforced
Type	String
Default	`|`
Bootstrap property	No
Required property	No
Restart required	No

Local configuration file	`AgentConfig.properties`
AM console	Tab: `Application` Title: `Not-Enforced Compound Rule Separator`

Invert Not-Enforced URIs

> **Note**
>
> The use of this property is NOT recommended.

When `true`, enforce policy for the URIs and patterns specified by the Not-Enforced URIs property, instead of allowing access to them without authentication.

For security considerations, do not enable this property. Instead, ForgeRock recommends using the NOT keyword to invert specific rules in the Not-Enforced URIs.

Property name	`org.forgerock.agents.notenforced.uri.invert.enabled`

Aliases	`com.sun.identity.agents.config.notenforced.uri.invert`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.notenforced.uri.invert.enabled`
	Introduced in Java Agent 5.6

Function	Not-enforced
Type	Boolean: `true` returns true; all other strings return `false`.
Default	`false`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Application` Title: `Invert Not-Enforced URIs` Legacy title: `Invert Not Enforced URIs`

Max Entries in Not-Enforced URI Cache

The maximum number of cached resource URLs that are matched by a not-enforced rule (inverted or not inverted).

Property name	`org.forgerock.agents.notenforced.uri.cache.size`

Aliases	`com.sun.identity.agents.config.notenforced.uri.cache.size`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.notenforced.uri.cache.size`
	Introduced in Java Agent 5.6

Function	Not-enforced
Type	Integer
Default	`10000`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Application` Title: `Max Entries in Not-Enforced URI Cache` Legacy title: `Not Enforced URIs Cache Size`

Enable Not-Enforced IP Cache

> **Note**
>
> The use of this property is NOT recommended.

When `true`, the agent caches evaluations of the Not-Enforced Client IP List.

Enable this setting if you are configuring many rules.

Property name	`org.forgerock.agents.notenforced.ip.cache.enabled`

Aliases	`org.forgerock.agents.notenforced.ip.cache.enabled`
	Introduced in Java Agent 5.6
	`com.sun.identity.agents.config.notenforced.ip.cache.enable`
	Introduced in Java Agent 5.0
	Recognized from AM 6

Function	Not-enforced
Type	Boolean: `true` returns true; all other strings return `false`.

Default	`true`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Application` Title: `Enable Not-Enforced IP Cache` Legacy title: `Not Enforced IP Cache Flag`

Invert Not-Enforced IPs

When `true`, enforce policy for the IPs specified by the Not-Enforced Client IP List property, instead of allowing access to them without authentication.

For security considerations, do not enable this property. Instead, ForgeRock recommends using the NOT keyword to invert specific rules in the Not-Enforced Client IP List.

Property name	`org.forgerock.agents.notenforced.ip.invert.enabled`

Aliases	`org.forgerock.agents.notenforced.ip.invert.enabled`
	Introduced in Java Agent 5.6
	`com.sun.identity.agents.config.notenforced.ip.invert`
	Introduced in Java Agent 5.0
	Recognized from AM 6

Function	Not-enforced
Type	Boolean: `true` returns true; all other strings return `false`.
Default	`false`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Application` Title: `Invert Not-Enforced IPs` Legacy title: `Invert Not Enforced IPs`

Max Entries in Not-Enforced IP Cache

The maximum number of cached IP addresses that are matched by a not-enforced rule (inverted or not inverted).

Property name	`org.forgerock.agents.notenforced.ip.cache.size`

Aliases	`org.forgerock.agents.notenforced.ip.cache.size`
	Introduced in Java Agent 5.6
	`com.sun.identity.agents.config.notenforced.ip.cache.size`
	Introduced in Java Agent 5.0
	Recognized from AM 6

Function	Not-enforced
Type	Integer
Default	`1000`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Application` Title: `Max Entries in Not-Enforced IP Cache` Legacy title: `Not Enforced IP Cache Size`

Container Parameter Encoding

The character encoding used by the agent when encoding extended characters in the HTTP query parameters of not-enforced rules.

Property name	`org.forgerock.agents.container.param.encoding`
Aliases	`org.forgerock.agents.container.param.encoding` Introduced in Java Agent 5.9.1
Function	Container, Not-enforced
Type	String
Default	`ISO-8859-1`
Bootstrap property	No

Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`

Notifications

Enable Notifications of Agent Configuration Change

Flag to indicate whether the agent subscribes to WebSocket notifications from AM for configuration changes. This property applies only the agent profile is stored in AM's configuration data store.

Property name	`org.forgerock.agents.config.change.notifications.enabled`

Aliases	`org.forgerock.agents.config.change.notifications.enabled`
	Introduced in Java Agent 5.6
	`com.sun.identity.agents.config.change.notification.enable`
	Introduced in Java Agent 5.0
	Recognized from AM 6

Function	Notifications
Type	Boolean: `true` returns true; all other strings return `false`.
Default	`true`
Bootstrap property	Yes
Required property	No
Restart required	No
Local configuration file	`AgentBootstrap.properties`
AM console	Tab: `Global` Title: `Enable Notifications of Agent Configuration Change` Legacy title: `Agent Configuration Change Notification`

Enable Notification of Session Logout (deprecated)

Use Enable Notification of Session Logout instead of this property.

Flag to indicate whether the agent polls AM for session status updates.

Default: `false`

Property name	`com.iplanet.am.session.client.polling.enabled`
Deprecated in	`5.6`

Aliases	`com.iplanet.am.session.client.polling.enabled`
	Introduced in Java Agent 5.6
	`com.iplanet.am.session.client.polling.enable`
	Introduced in Java Agent 5.0
	Recognized from AM 6

Function	Notifications
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`

Enable Notification of Policy Changes

Flag to indicate whether the agent subscribes to WebSocket notifications from AM for policy changes.

Property name	`org.forgerock.agents.policy.change.notifications.enabled`

Aliases	`org.forgerock.agents.policy.change.notifications.enabled`
	Introduced in Java Agent 5.6
	`com.sun.identity.agents.notification.enabled`
	Introduced in Java Agent 5.0
	Recognized from AM 6

Function	Notifications
Type	Boolean: `true` returns true; all other strings return `false`.
Default	`true`
Bootstrap property	Yes
Required property	No
Restart required	No
Local configuration file	`AgentBootstrap.properties`

AM console	Tab: `AM Services` Title: `Enable Notification of Policy Changes`

Enable Notification of Session Logout

Flag to indicate whether the agent subscribes to WebSocket notifications from AM for session logout.

Default: `true`

Property name	`org.forgerock.agents.session.change.notifications.enabled`
Aliases	`org.forgerock.agents.session.change.notifications.enabled` Introduced in Java Agent 5.6 Recognized from AM 7
Function	Notifications
Type	Boolean: `true` returns true; all other strings return `false`.
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Global (from AM 7)` Title: `Enable Notification of Session Logout`

POST data preservation

POST Data Preservation Max HTML Form Size

The maximum size of all name/value pairs submitted in an HTML form during POST data preservation. Set to zero to remove the limit.

This property is valid only when Enable POST Data Preservation is `true`.

Property name	`org.forgerock.agents.pdp.parameter.limit.bytes`
Aliases	`org.forgerock.agents.pdp.parameter.limit.bytes` Introduced in Java Agent 2023.11 Recognized from AM 7.1

Function	POST data preservation
Type	Integer
Default	`104857600`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`

POST Data Preservation Directory Sweep Interval

The interval in seconds at which the POST Data Preservation File Directory is swept for expired files. Files expire after the time defined by POST Data Preservation Cache TTL.

This property is valid only when Enable POST Data Preservation and POST Data Preservation in Files or Cache are `true`, and POST Data Preservation File Directory is a valid directory.

Property name	`org.forgerock.agents.pdp.directory.sweep.seconds`
Aliases	`org.forgerock.agents.pdp.directory.sweep.seconds` Introduced in Java Agent 5.10.0
Function	POST data preservation
Type	Integer
Default	`10`
Bootstrap property	Yes
Required property	No
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentBootstrap.properties`

Missing POST Data Preservation Entry URI Map

A map of URLs to which the agent redirects when the POST data preservation cache entry is discarded due to a cache timeout. The URL is expected to be a page explaining what has happened.

Property name	`org.forgerock.agents.pdp.noentry.url.map`

Aliases	`org.forgerock.agents.pdp.noentry.url.map`
	Introduced in Java Agent 5.6
	`com.sun.identity.agents.config.postdata.preserve.cache.noentry.url`
	Introduced in Java Agent 5.0
	Recognized from AM 6

Function	POST data preservation
Type	Map • Keys: web application • Values: URL of page explaining the PDP entry was discarded
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Advanced` Title: `Missing POST Data Preservation Entry URI Map` Legacy title: `Missing PDP entry URI`

POST Data Preservation Cache TTL

The interval in minutes at which entries in the POST data preservation cache timeout and are purged.

If this property and POST Data Preservation Cache TTL in Milliseconds (deprecated) are set, POST Data Preservation Cache TTL in Milliseconds (deprecated) takes precedence.

Property name	`org.forgerock.agents.pdp.cache.ttl.minutes`
Aliases	`org.forgerock.agents.pdp.cache.ttl.minutes` Introduced in Java Agent 5.6 Recognized from AM 7
Function	POST data preservation
Type	Integer
Default	`5`
Bootstrap property	Yes

Required property	No
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentBootstrap.properties`
AM console	Tab: `Advanced (from AM 7)` Title: `POST Data Preservation Cache TTL` Legacy title: `PDP Cache TTL in Minutes`

POST Data Preservation Sticky Session Key Value

A name/value pair separated by `=`, as follows:

When POST Data Preservation Sticky Session Mode is `URL`, this property sets the query parameter name and value.

When POST Data Preservation Sticky Session Mode is `Cookie`, this property sets the cookie name and value.

Property name	`org.forgerock.agents.pdp.sticky.session.value`

Aliases	`com.sun.identity.agents.config.postdata.preserve.stickysession.value`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.pdp.sticky.session.value`
	Introduced in Java Agent 5.6

Function	POST data preservation
Type	String
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Advanced` Title: `POST Data Preservation Sticky Session Key Value` Legacy title: `PDP Stickysession key-value`

POST Data Preservation Sticky Session Mode

Property name	`org.forgerock.agents.pdp.sticky.session.mode`

Aliases	`org.forgerock.agents.pdp.sticky.session.mode`
	Introduced in Java Agent 5.6
	`com.sun.identity.agents.config.postdata.preserve.stickysession.mode`
	Introduced in Java Agent 5.0
	Recognized from AM 6

Function	POST data preservation
Supported settings	***COOKIE*** The sticky session mode is sent as a cookie (name/value specified by "POST Data Preservation Sticky Session Key Value" (org.forgerock.agents.pdp.sticky.session.value)). ***URL*** The sticky session mode is sent as an HTTP parameter name/value pair (specified by "POST Data Preservation Sticky Session Key Value" (org.forgerock.agents.pdp.sticky.session.value)).
Default	`URL`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Advanced` Title: `POST Data Preservation Sticky Session Mode` Legacy title: `PDP Stickysession mode`

Enable POST Data Preservation

When `true`, unauthenticated POST data is stored before redirecting to the login screen, then auto-submitted after successful authentication.

Property name	`org.forgerock.agents.post.data.preservation.enabled`

Aliases	`org.forgerock.agents.post.data.preservation.enabled`
	Introduced in Java Agent 5.6
	`com.sun.identity.agents.config.postdata.preserve.enable`
	Introduced in Java Agent 5.0
	Recognized from AM 6

Function	POST data preservation

Type	Boolean: `true` returns true; all other strings return `false`.
Default	`false`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Advanced` Title: `Enable POST Data Preservation` Legacy title: `Post Data Preservation enabled`

POST Data Preservation in Files or Cache

When Enable POST Data Preservation is `true`, this property determines how POST data is saved:

`true`: Save POST data to files

`false` (default): Save POST data to the in-memory cache

Files are stored in the POST Data Preservation File Directory. Expired files are removed at the interval given by POST Data Preservation Directory Sweep Interval.

Property name	`org.forgerock.agents.pdp.use.filesystem.enabled`
Aliases	`org.forgerock.agents.pdp.use.filesystem.enabled` Introduced in Java Agent 5.10.0
Function	POST data preservation
Type	Boolean: `true` returns true; all other strings return `false`.
Default	`false`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`

Pre-Authn and Post Data Preservation Cookie Signing Value

The key to sign pre-authentication cookies and POST data preservation cookies.

The key is set during installation, when the agent requests the path to a file containing the cookie signing key, and then uses the key to set the cookie signing value in the `AgentKey.properties` file. For information about how to set or change the key after installation, see Rotate cookie signing keys.

For security, you are recommended to configure cookie signing. The agent does not sign cookies when:

- The path to the signing key is left blank during installation.
- The signing key in the `AgentKey.properties` file is less than 64-characters long.

Property name	`org.forgerock.agents.cookie.signing.value`
Aliases	`org.forgerock.agents.cookie.signing.value` Introduced in Java Agent 5.10.0
Function	Cookie, POST data preservation, Pre-authentication
Type	String
Bootstrap property	No
Required property	No
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentKey.properties`

POST Data Preservation Storage Size

The maximum number of megabytes allocated to store POST data. When the maximum is reached, old entries are discarded.

Use this property to mitigate the risk of DDoS attacks.

This property takes precedence over Max Entries in POST Data Preservation Storage.

Property name	`org.forgerock.agents.pdp.cache.total.size.mb`

Aliases	`org.forgerock.agents.pdp.cache.total.size.mb`
	Introduced in Java Agent 5.6
	`org.forgerock.openam.agents.config.postdata.preserve.cache.entry.max.total.size.mb`
	Introduced in Java Agent 5.6
	Recognized from AM 6

Function	POST data preservation

Type	Integer
Default	``
Bootstrap property	Yes
Required property	No
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentBootstrap.properties`
AM console	Tab: `Advanced` Title: `POST Data Preservation Storage Size` Legacy title: `PDP Maximum Cache Size`

Max Entries in POST Data Preservation Storage

When POST Data Preservation in Files or Cache is `false`, the maximum number of entries in the POST data preservation storage cache.

When POST Data Preservation in Files or Cache is `true`, the maximum number of files in the POST Data Preservation File Directory.

When the maximum is reached, old entries are discarded.

Use this property to mitigate the risk of DoS attacks.

POST Data Preservation Storage Size takes precedence over this property.

Property name	`org.forgerock.agents.pdp.cache.size`

Aliases	`org.forgerock.agents.pdp.cache.size`
	Introduced in Java Agent 5.6
	`org.forgerock.openam.agents.config.postdata.preserve.cache.entry.max.entries`
	Introduced in Java Agent 5.6
	Recognized from AM 6

Function	POST data preservation
Type	Integer
Default	`1000`
Bootstrap property	Yes
Required property	No

Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentBootstrap.properties`
AM console	Tab: `Advanced` Title: `Max Entries in POST Data Preservation Storage` Legacy title: `PDP Maximum Number of Cache Entries`

POST Data Preservation File Directory

When Enable POST Data Preservation and POST Data Preservation in Files or Cache are `true`, this property defines the directory in which POST data preservation data files are saved.

The agent must be able to access the directory. If the directory does not exist, or the agent cannot access it, Enable POST Data Preservation is set to `false`.

Default: `/path/to/java_agents/agent_type/Agent_n/pdp`.

Property name	`org.forgerock.agents.pdp.directory`
Aliases	`org.forgerock.agents.pdp.directory` Introduced in Java Agent 5.10.0
Function	POST data preservation
Type	String
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`

Post Data Preservation Cookie Name

The name of the Post Data Preservation cookie. This cookie maintains the security of the data in unauthenticated POST requests. It contains a unique one-time code which is cross-checked against the stored data making it extremely difficult for malicious actors to replay the stored data for other users.

Since Java Agent 5.10, there is the option of creating one cookie for all concurrent PDP requests, or alternatively to have one uniqely named cookie per request.

If you have tests in your environment that make multiple PDP requests to the agent, you may find intermittent failures as browsers can limit how many cookies they handle.

Configure the cookie name as follows:

- To use one cookie for all concurrent PDP requests to AM, configure as a string. For example, `org.forgerock.agents.pdp.cookie.name=cookie-name`.
- To use one cookie for each authentication request to AM, configure as `%n` before, in the middle, or after a string. When the agent creates the cookie, it substitutes `%n` for a unique identifier. For example:
  - `org.forgerock.agents.pdp.cookie.name=%n`
  - `org.forgerock.agents.pdp.cookie.name=%n-cookie-name`
  - `org.forgerock.agents.pdp.cookie.name=cookie-%n-name`
  - `org.forgerock.agents.pdp.cookie.name=cookie-name-%n`

The agent compresses and then signs the cookie.

Property name	`org.forgerock.agents.pdp.cookie.name`
Aliases	`org.forgerock.agents.pdp.cookie.name` Introduced in Java Agent 5.10.0 Recognized from AM 7.1
Function	Cookie, POST data preservation
Type	String
Default	`PDP_Nonce`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`

POST Data Preservation Cache TTL in Milliseconds (deprecated)

Specifies the POST data preservation cache timeout in milliseconds.

Use POST Data Preservation Cache TTL instead of this property.

If this property and POST Data Preservation Cache TTL are set, this property takes precedence.

Property name	`com.sun.identity.agents.config.postdata.preserve.cache.entry.ttl`
Deprecated in	`5.6`

Aliases	`com.sun.identity.agents.config.postdata.preserve.cache.entry.ttl` Introduced in Java Agent 5.0
Function	POST data preservation
Type	Integer
Default	`-2147483648`
Bootstrap property	Yes
Required property	No
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentBootstrap.properties`

Policy enforcement

POST Parameter List for URL Policy Env

The list of HTTP POST request parameters whose names and values the agent sets in the environment map for URL policy evaluation by the AM server.

Property name	`org.forgerock.agents.continuous.security.post.list`

Aliases	`com.sun.identity.agents.config.policy.env.post.param`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.continuous.security.post.list`
	Introduced in Java Agent 5.7

Function	Policy enforcement
Type	List
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `AM Services` Title: `POST Parameter List for URL Policy Env` Legacy title: `URL Policy Env POST Parameters`

Max Entries in Policy Cache per Session

The maximum number of policy evaluation entries allowed in the policy evaluation cache for each session.

The number of policy evaluation results that can be stored is this property multiplied by the value of Max Sessions in Policy Cache.

Property name	`org.forgerock.agents.policy.cache.per.session.size`

Aliases	`com.sun.identity.policy.client.cacheResultsPerUserCap`
	Introduced in Java Agent 5.0
	`org.forgerock.agents.policy.cache.per.session.size`
	Introduced in Java Agent 5.6
	Recognized from AM 7

Function	Policy enforcement
Type	Integer
Default	`50`
Bootstrap property	Yes
Required property	No
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentBootstrap.properties`
AM console	Tab: `Advanced (from AM 7)` Title: `Max Entries in Policy Cache per Session` Legacy title: `Policy Cache Per User`

Restrict to Realm Map

A map to restrict access to the specified web application to users authenticated in the specified realm.

Property name	`org.forgerock.agents.restrict.to.realm.map`
Aliases	`org.forgerock.agents.restrict.to.realm.map` Introduced in Java Agent 5.6.2.1 Recognized from AM 7
Function	Policy enforcement

Type	Map • Keys: web application • Values: allowed realms as CSV
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `AM Services (from AM 7)` Title: `Restrict to Realm Map` Legacy title: `Restrict To Realm`

Enable Composite Advice Encoding

When `true`, composite advices are base64 URL-encoded before being sent to custom login endpoints. Use this property to increase security, and protect against cross-site scripting attacks.

Property name	`org.forgerock.agents.advice.b64.url.encode.enabled`

Aliases	`com.forgerock.agents.advice.b64.url.encode`
	Introduced in Java Agent 5.6.2.1
	Recognized from AM 7
	`org.forgerock.agents.advice.b64.url.encode.enabled`
	Introduced in Java Agent 5.6.2.1
	`org.forgerock.agents.advice.b64.url.encode`
	Introduced in Java Agent 5.6.2.1

Function	Policy enforcement
Type	Boolean: `true` returns true; all other strings return `false`.
Default	`false`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`

AM console	Tab: `AM Services (from AM 7)` Title: `Enable Composite Advice Encoding` Legacy title: `Composite Advice Encode`

Max Sessions in Policy Cache

The maximum number of sessions (distinct users) that can be stored in the policy evaluation cache at any time.

Property name	`org.forgerock.agents.policy.cache.session.size`

Aliases	`org.forgerock.agents.policy.cache.session.size`
	Introduced in Java Agent 5.6
	Recognized from AM 7
	`com.sun.identity.policy.client.cachedSessionCap`
	Introduced in Java Agent 5.0

Function	Policy enforcement
Type	Integer
Default	`5000`
Bootstrap property	Yes
Required property	No
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentBootstrap.properties`
AM console	Tab: `Advanced (from AM 7)` Title: `Max Sessions in Policy Cache` Legacy title: `Policy Cache Size`

Enable Policy Evaluation in User Authentication Realm

When `true`, perform policy evaluation in the realm to which the user is authenticated, and ignore the value in Policy Evaluation Realm Map.

Use this property for web applications that dynamically set the realm for authentication.

Property name	`org.forgerock.agents.user.realm.overrides.policy.evaluation.realm.enabled`

Aliases	`org.forgerock.agents.user.realm.overrides.policy.evaluation.realm.enabled` Introduced in Java Agent 5.7.1 Recognized from AM 7.1
Function	Policy enforcement
Type	Boolean: `true` returns true; all other strings return `false`.
Default	`false`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `AM Services (from AM 7.1)` Title: `Enable Policy Evaluation in User Authentication Realm` Legacy title: `Perform Policy Evaluation in User Authenticated Realm`

GET Parameter List for URL Policy Env

The list of HTTP GET request parameters whose names and values the agent sets in the environment map for URL policy evaluation by the AM server.

Property name	`org.forgerock.agents.continuous.security.get.list`

Aliases	`com.sun.identity.agents.config.policy.env.get.param`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.continuous.security.get.list`
	Introduced in Java Agent 5.7

Function	Policy enforcement
Type	List
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`

AM console	Tab: `AM Services` Title: `GET Parameter List for URL Policy Env` Legacy title: `URL Policy Env GET Parameters`

Policy Cache TTL

The interval in minutes at which entries in the policy cache time out and are purged.

Property name	`org.forgerock.agents.policy.cache.ttl.minutes`

Aliases	`org.forgerock.agents.policy.cache.ttl.minutes`
	Introduced in Java Agent 5.6
	`com.sun.identity.agents.polling.interval`
	Introduced in Java Agent 5.0
	Recognized from AM 7

Function	Policy enforcement
Type	Integer
Default	`3`
Bootstrap property	Yes
Required property	No
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentBootstrap.properties`
AM console	Tab: `Advanced` Title: `Policy Cache TTL`

JSession Parameter List for URL Policy Env

The list of HTTP session attributes whose names and values the agent sets in the environment map for URL policy evaluation by the AM server.

Property name	`org.forgerock.agents.continuous.security.http.session.list`

Aliases	`org.forgerock.agents.continuous.security.http.session.list`
	Introduced in Java Agent 5.7
	`com.sun.identity.agents.config.policy.env.jsession.param`
	Introduced in Java Agent 5.0
	Recognized from AM 6

Function	Policy enforcement
Type	List
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `AM Services` Title: `JSession Parameter List for URL Policy Env` Legacy title: `URL Policy Env jsession Parameters`

Policy Evaluation Realm Map

The name of an AM realm to use for policy evaluations. Different web applications can use different policy realms.

> **Note**
>
> To prevent errors, use only characters in the standard ASCII set. The agent automatically percent-encodes characters in the extended ASCII set, however, the name must be manually encoded in AM.

Property name	`org.forgerock.agents.policy.evaluation.realm.map`

Aliases	`org.forgerock.openam.agents.config.policy.evaluation.realm`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.policy.evaluation.realm.map`
	Introduced in Java Agent 5.6.2.1

Function	Policy enforcement
Type	Map • Keys: web application • Values: realm
Default	`/`
Bootstrap property	No
Required property	No
Restart required	No

Local configuration file	`AgentConfig.properties`
AM console	Tab: `AM Services` Title: `Policy Evaluation Realm Map` Legacy title: `Policy Evaluation Realm`

Policy Set Map

The name of an AM policy set to use for policy evaluations. Different web applications can use a different policy set in their chosen realm.

> **Note**
>
> To prevent errors, use only characters in the standard ASCII set. The agent automatically percent-encodes characters in the extended ASCII set, however, the name must be manually encoded in AM.

The following example causes AM to use `mypolicyset` to evaluate policies for all web applications:

```
org.forgerock.agents.policy.set.map=mypolicyset
```

The following example causes AM to use `mypolicyset` to evaluate policies for `mywebapp`. For all other web applications, AM uses `iPlanetAMWebAgentService`:

```
org.forgerock.agents.policy.set.map[mywebapp]=mypolicyset
```

Property name	`org.forgerock.agents.policy.set.map`

Aliases	`org.forgerock.openam.agents.config.policy.evaluation.application`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.policy.set.map`
	Introduced in Java Agent 5.6.2.1

Function	Policy enforcement
Type	Map • Keys: web application • Values: policy set
Default	`iPlanetAMWebAgentService`
Bootstrap property	No
Required property	No
Restart required	No

Local configuration file	`AgentConfig.properties`
AM console	Tab: `AM Services` Title: `Policy Set Map` Legacy title: `Policy Set`

Pre-authentication

Pre-Authn and Post Data Preservation Cookie Signing Value

The key to sign pre-authentication cookies and POST data preservation cookies.

The key is set during installation, when the agent requests the path to a file containing the cookie signing key, and then uses the key to set the cookie signing value in the `AgentKey.properties` file. For information about how to set or change the key after installation, see Rotate cookie signing keys.

For security, you are recommended to configure cookie signing. The agent does not sign cookies when:

- The path to the signing key is left blank during installation.
- The signing key in the `AgentKey.properties` file is less than 64-characters long.

Property name	`org.forgerock.agents.cookie.signing.value`
Aliases	`org.forgerock.agents.cookie.signing.value` Introduced in Java Agent 5.10.0
Function	Cookie, POST data preservation, Pre-authentication
Type	String
Bootstrap property	No
Required property	No
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentKey.properties`

Max Age of Pre-Authentication Cookie

The maximum age in seconds of the pre-authentication cookie configured in Pre-Authentication Cookie Name.

Property name	`org.forgerock.agents.authn.cookie.max.age.seconds`

Aliases	`org.forgerock.agents.authn.cookie.max.age.seconds` Introduced in Java Agent 5.6.3.0 Recognized from AM 7
Function	Cookie, Pre-authentication
Type	Integer
Default	`600`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Global (from AM 7)` Title: `Max Age of Pre-Authentication Cookie` Legacy title: `Pre-Authenticated Cookie Max Age`

Pre-Authentication Cookie Name

The name of the pre-authentication cookie. This cookie tracks the progress of authentication with AM, and protects requests from replay attacks. It contains the following information:

- URL of the original request
- HTTP mode
- Secure ID (subsequently baked into the nonce of the returned JWT)
- Relevant ACR information
- Transaction ID
- Expiry time configured by Max Age of Pre-Authentication Cookie

(Before Java Agent 5.7), 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.

(Java Agent 5.7 and later versions) The agent can optionally create a cookie for each authentication request to AM. In some environments, this creates 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 can limit how many cookies they handle.

Configure the cookie name as follows:

- To use one cookie for all concurrent authentication requests to AM, configure as a string. For example, `org.forgerock.agents.authn.cookie.name=cookie-name`.
- To use one cookie for each authentication request to AM, configure as `%n`, or as `%n` before, in the middle of, or after a string. When the agent creates the cookie, it translates the string `%n` into a unique identifier. For example:
  - `org.forgerock.agents.authn.cookie.name=%n`
  - `org.forgerock.agents.authn.cookie.name=%n-cookie-name`
  - `org.forgerock.agents.authn.cookie.name=cookie-%n-name`
  - `org.forgerock.agents.authn.cookie.name=cookie-name-%n`

The agent compresses and then signs the cookie.

Property name	`org.forgerock.agents.authn.cookie.name`

Aliases	`com.sun.identity.agents.config.cdsso.cookie.name`
	Introduced in Java Agent 5.0
	Recognized from AM 7
	`org.forgerock.agents.authn.cookie.name`
	Introduced in Java Agent 5.6

Function	Cookie, Pre-authentication
Type	String
Default	`amFilterCDSSORequest`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Global (from AM 7)` Title: `Pre-Authentication Cookie Name` Legacy title: `Pre-Authenticated Cookie Name`

Profile

Location of Agent Configuration Repository

The location of the agent configuration.

Property name	`org.forgerock.agents.config.location`

Aliases	`com.sun.identity.agents.config.repository.location`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.config.location`
	Introduced in Java Agent 5.6

Function	Profile, Required
Supported settings	***LOCAL*** The agent configuration is read from AgentConfiguration.properties (only). ***REMOTE*** The agent configuration is downloaded from AM.
Default	`REMOTE`
Bootstrap property	Yes
Required property	Yes - If this property is missing, the agent fails to start
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentBootstrap.properties`
AM console	Tab: `Global` Title: `Location of Agent Configuration Repository`

JWT Cookie Domain List

A list of domains in which the agent attempts to creates JWT cookies:

- If the list is empty, the agent creates cookies only in its own domain.
- If the agent is running behind a browser, it can create cookies only in its own domain.
- If the agent is running behind a proxy, it should be able to create cookies in any required domains.

Default: Empty

Property name	`org.forgerock.agents.jwt.cookie.domain.list`

Aliases	`com.sun.identity.agents.config.cdsso.domain`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.jwt.cookie.domain.list`
	Introduced in Java Agent 5.6

Function	Profile
Type	List
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `SSO (from AM 7.1)` Title: `JWT Cookie Domain List` Legacy title: `CDSSO Domain List`

JWT Cache TTL

The interval in minutes at which entries in the JWT cache timeout and are purged.

Parsing JWTs is a CPU intensive process. As all JWTs in the cache have already been parsed, consider using a long timeout for this cache.

Property name	`org.forgerock.agents.jwt.cache.ttl.minutes`

Aliases	`org.forgerock.openam.agents.config.jwt.cache.ttl.minutes`
	Introduced in Java Agent 5.0
	Recognized from AM 7
	`org.forgerock.agents.jwt.cache.ttl.minutes`
	Introduced in Java Agent 5.6

Function	Profile
Type	Integer
Default	`30`
Bootstrap property	Yes
Required property	No
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentBootstrap.properties`
AM console	Tab: `Advanced (from AM 7)` Title: `JWT Cache TTL`

Max Entries in JWT Cache

The maximum number of entries in the JWT cache.

Property name	`org.forgerock.agents.jwt.cache.size`

Aliases	`org.forgerock.agents.jwt.cache.size`
	Introduced in Java Agent 5.6
	`org.forgerock.openam.agents.config.jwt.cache.size`
	Introduced in Java Agent 5.0
	Recognized from AM 7

Function	Profile
Type	Integer
Default	`5000`
Bootstrap property	Yes
Required property	No
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentBootstrap.properties`
AM console	Tab: `Advanced (from AM 7)` Title: `Max Entries in JWT Cache` Legacy title: `JWT Cache Size`

JWT Cookie Name

The name of the cookie that holds the OIDC JWT on the user's browser.

Before changing the name of this cookie, consider the following points:

- This cookie is only used by the agent and is never presented to AM.
- The cookie name must be unique in the cookies the user's browser receives. For example, do not set the JWT cookie name to `iPlanetDirectoryPro`, which is the default name of the AM session cookie.

If the agent does not find the cookie named by JWT Cookie Name, authentication fails. The user can only access resources that are available through not-enforced rules.

Property name	`org.forgerock.agents.jwt.cookie.name`

Aliases	`org.forgerock.openam.agents.config.jwt.name`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.jwt.cookie.name`
	Introduced in Java Agent 5.6

Function	Profile
Type	String
Default	`am-auth-jwt`
Bootstrap property	Yes
Required property	No
Restart required	No
Local configuration file	`AgentBootstrap.properties`
AM console	Tab: `Global` Title: `JWT Cookie Name`

Agent Profile Realm

The realm in which the agent profile is defined.

When Enable Policy Evaluation in User Authentication Realm is `true`, AM uses this realm to evaluate polices for policy decision requests from the agent.

Property name	`org.forgerock.agents.agent.profile.realm`

Aliases	`org.forgerock.agents.agent.profile.realm`
	Introduced in Java Agent 5.6
	`com.sun.identity.agents.config.organization.name`
	Introduced in Java Agent 5.0

Function	Profile, Required
Type	String
Default	`/`
Bootstrap property	Yes
Required property	Yes - If this property is missing, the agent fails to start
Restart required	Yes - Restart the container after changing the property

Local configuration file	`AgentBootstrap.properties`

Exchanged SSO Token Cache TTL

The interval in minutes at which entries in the SSO token exchange cache timeout and are purged.

The exchanged JWT is cached against the relevant SSO token. If the same SSO token is presented again, before the cache entry expires, the agent does not need to exchange the token again, but retrieves the one stored in its cache.

Because exchanging SSO tokens for JWTs is an expensive process, previously exchanged SSO tokens are cached. When an entity is unable to permanently store its JWT in a cookie, calls to AM can be avoided.

Property name	`org.forgerock.agents.sso.exchange.cache.ttl.minutes`
Aliases	`org.forgerock.agents.sso.exchange.cache.ttl.minutes` Introduced in Java Agent 5.6.2.1 Recognized from AM 7
Function	Profile
Type	Integer
Default	`5`
Bootstrap property	Yes
Required property	No
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentBootstrap.properties`
AM console	Tab: `Advanced (from AM 7)` Title: `Exchanged SSO Token Cache TTL` Legacy title: `Exchanged SSO Token Cache Time to Live`

Configuration Reload Interval

When the Location of Agent Configuration Repository is `LOCAL`, this is the number of seconds after which the agent reloads its configuration if it has been changed since it was last read.

Property name	`org.forgerock.agents.config.reload.seconds`

Aliases	`com.sun.identity.agents.config.load.interval`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.config.reload.seconds`
	Introduced in Java Agent 5.6

Function	Profile
Type	Integer
Default	`0`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Global` Title: `Configuration Reload Interval`

Agent Profile Name

The profile name used to fetch agent configuration data from AM, to evaluate policies for users, retrieve session info, and so on.

Default: Empty

Property name	`org.forgerock.agents.profile.name`

Aliases	`com.sun.identity.agents.app.username`
	Introduced in Java Agent 5.0
	`com.sun.identity.agents.config.profilename`
	Introduced in Java Agent 5.0
	`org.forgerock.agents.profile.name`
	Introduced in Java Agent 5.6

Function	Profile, Required
Type	String
Bootstrap property	Yes
Required property	Yes - If this property is missing, the agent fails to start
Restart required	Yes - Restart the container after changing the property

Local configuration file	`AgentBootstrap.properties`

Enable Configuration Lock

When `true`, an agent restart is required to allow configuration changes, even for hot-swappable parameters.

Property name	`org.forgerock.agents.lock.config.enabled`

Aliases	`com.sun.identity.agents.config.lock.enable`
	Introduced in Java Agent 5.0
	`org.forgerock.agents.lock.config.enabled`
	Introduced in Java Agent 5.6

Function	Profile
Type	Boolean: `true` returns true; all other strings return `false`.
Default	`false`
Bootstrap property	Yes
Required property	No
Restart required	No
Local configuration file	`AgentBootstrap.properties`

Profile Attribute Map

Map the value of an AM profile attribute to one or more HTTP headers, HTTP cookies, or request attributes, depending on the value of Profile Attribute Fetch Mode.

- Map key: The name of an AM profile attribute
- Map value: The name of one or more HTTP headers, HTTP cookies, or request attributes

The user profile can be stored in LDAP or any other arbitrary data store

Consider the following points for HTTP 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 and the cookie value is different from the mapped value, the agent overwrites the cookie.
- If an HTTP cookie with the mapped name already exists and the cookie value is the same as the mapped value, the agent does nothing.
- If the profile attribute name (key) does not exist, the agent does not create the HTTP cookie.

- The agent does not automatically clear cookies. To prevent a build up of cookies, consider adding them to the Reset Cookie List.

Consider the following points for HTTP headers:

- If an HTTP header with the mapped name does not exist, the agent creates it.
- If an HTTP header with the mapped name already exists, the agent does not overwrite it but simply appends information to the header.
- If the profile attribute name (key) does not exist, the agent does not create the HTTP header.
- When an HTTP header name is used in a request header, it is prefixed by `HTTP_`. The agent automatically changes lower case letters to upper case, and hyphens (`-`) to underscores (`_`). For example, `CUSTOM-userid` becomes `HTTP_CUSTOM_USERID`.

Format: `profile attribute = HEADER-NAME(S)`, `profile attribute = COOKIE-NAME(S)`, `profile attribute = REQUEST-ATTRIBUTE(S)`

Default: Empty

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`

Example: `[cn]=HEADER-1|HEADER-2`

Property name	`org.forgerock.agents.profile.attribute.map`

Aliases	`com.sun.identity.agents.config.profile.attribute.mapping`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.profile.attribute.map`
	Introduced in Java Agent 5.6

Function	Profile
Type	Map • Keys: source response profile name • Values: one or more target response profile names
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`

AM console	Tab: `Application` Title: `Profile Attribute Map` Legacy title: `Profile Attribute Mapping`

Max Entries in SSO Exchange Cache

The maximum number of entries in the SSO exchange cache, used when SSO tokens are exchanged for JWTs.

When the maximum is reached, the oldest records are overwritten.

Property name	`org.forgerock.agents.sso.exchange.cache.size`
Aliases	`org.forgerock.agents.sso.exchange.cache.size` Introduced in Java Agent 5.6.2.1 Recognized from AM 7
Function	Profile
Type	Integer
Default	`100`
Bootstrap property	Yes
Required property	No
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentBootstrap.properties`
AM console	Tab: `Advanced (from AM 7)` Title: `Max Entries in SSO Exchange Cache` Legacy title: `Exchanged SSO Token Cache Size`

Profile Attribute Fetch Mode

Map the name of an AM profile attribute specified in Profile Attribute Map as follows:

- NONE: Do not map
- HTTP_HEADER: Map the to the name of an HTTP profile header
- HTTP_COOKIE: Map to the name of an HTTP cookie
- REQUEST_ATTRIBUTE: Map to the name of an HTTP profile attribute

Property name	`org.forgerock.agents.profile.attribute.fetch.mode`

Aliases	`com.sun.identity.agents.config.profile.attribute.fetch.mode`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.profile.attribute.fetch.mode`
	Introduced in Java Agent 5.6

Function	Attributes, Cookie reset, Profile
Supported settings	***NONE*** No AM attributes are mapped. ***HTTP_HEADER*** The named AM attributes are mapped to headers in the outgoing HTTP response. ***REQUEST_ATTRIBUTE*** The named AM attributes are mapped to request attributes in the outgoing HTTP response. ***HTTP_COOKIE*** The named AM attributes are mapped to cookies in the outgoing HTTP response.
Default	`NONE`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Application` Title: `Profile Attribute Fetch Mode`

WebSocket Connection Interval

The interval in minutes at which WebSockets to AM are killed and reopened. This property helps ensure a balanced distribution of connections across the AM servers on the site.

Property name	`org.forgerock.agents.balance.websocket.interval.minutes`

Aliases	`org.forgerock.openam.agents.config.balance.websocket.connection.interval.in.minutes`
	Introduced in Java Agent 5.6
	Recognized from AM 6
	`org.forgerock.agents.balance.websocket.interval.minutes`
	Introduced in Java Agent 5.6

Function	Profile
Type	Integer
Default	`30`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Global` Title: `WebSocket Connection Interval`

Query parameter

Regex Remove Query Parameters List for Policy Evaluation

A list of regular expressions the agent uses to match query parameters to be removed from the incoming URL for policy evaluation and caching purposes. The property has the following format, with no spaces between values:

`[Domain[/path]]|parameter[,parameter…]`

Consider the following constraints when constructing your list of regular expressions:

- Add a comma (,) character at the beginning or the end of the list to remove all unnamed parameters. For example, `myapp.example.com/customers|,lang` would match both `lang` and any unnamed parameters.
- Consider creating multiple simple regular expressions instead of a single complicated one.
- The remaining parameters (those that do not match the list of parameters) are sorted alphabetically.

Examples:

`org.forgerock.agents.unwanted.http.url.params.regex.list[0]=myapp.example.com|b.*,gp(a|p|s),`

`org.forgerock.agents.unwanted.http.url.params.regex.list[1]=|.*`

The following incoming URL request that matches a rule such as `myapp.example.com/customers|,coun.*?`:

`http://myapp.example.com/customers?country=uk&=bristol&lang=en_GB&area=1343456`

It is cached by the agent as `http://myapp.example.com/customers?=bristol&lang=en_GB`, where both `country` and unnamed parameter are removed and the remaining parameters are sorted alphabetically.

Property name	`org.forgerock.agents.unwanted.http.url.params.regex.list`

Aliases	`org.forgerock.agents.unwanted.http.url.params.regex.list`
	Introduced in Java Agent 5.6
	`org.forgerock.openam.agents.config.conditional.unwanted.http.url.params.regexp`
	Introduced in Java Agent 5.6
	Recognized from AM 7
	`org.forgerock.agents.unwanted.http.url.params.regexp.list`
	Introduced in Java Agent 5.6

Function	Query parameter
Type	List
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Miscellaneous (from AM 7)` Title: `Regex Remove Query Parameters List for Policy Evaluation` Legacy title: `Regular Expression Remove Query Parameters`

Remove Query Parameters List for Policy Evaluation

A list of query parameters to be removed from the incoming URL for policy evaluation and caching.

The property has the following format, with no spaces between values:

`[Domain[/path]]|parameter[,parameter…]`

Consider the following constraints when constructing the list of parameters:

- Add a comma (,) character at the beginning or the end of the list to remove all unnamed parameters. The following example would match both `lang` and any unnamed parameters: `myapp.example.com/customers|,lang`
- Add the asterisk (*) character to the list to remove all parameters, including unnamed ones.
- The remaining parameters (those that do not match the list of parameters) are sorted alphabetically.

Examples:

`org.forgerock.agents.unwanted.http.url.param.list[0]=myapp.example.com/customers|location,lang`

`org.forgerock.agents.unwanted.http.url.param.list[1]=example.com/customers|*`

The following incoming URL request matches a rule such as `myapp.example.com/customers|,lang`:

`http://myapp.example.com/customers?country=uk&=bristol&lang=en_GB&area=1343456`

It is cached by the agent as `http://myapp.example.com/customers?area=1343456&country=uk`, where both `lang` and the unnamed parameter are removed and the rest of the parameters are sorted alphabetically.

Property name	`org.forgerock.agents.unwanted.http.url.param.list`

Aliases	`org.forgerock.agents.unwanted.http.url.param.list`
	Introduced in Java Agent 5.6
	`org.forgerock.openam.agents.config.conditional.unwanted.http.url.params`
	Introduced in Java Agent 5.6
	Recognized from AM 7

Function	Query parameter
Type	List
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Miscellaneous (from AM 7)` Title: `Remove Query Parameters List for Policy Evaluation` Legacy title: `Remove Query Parameters`

Regex Query Parameters List for Policy Evaluation

A list of regular expressions the agent uses to match query parameters, for policy evaluation and caching.

The property has the following format, with no spaces between values:

`[Domain[/path]]|regexp[,regexp,…]`

Property name	`org.forgerock.agents.wanted.http.url.params.regexp.list`

Aliases	`org.forgerock.agents.wanted.http.url.params.regexp.list`
	Introduced in Java Agent 5.6
	`org.forgerock.openam.agents.config.conditional.wanted.http.url.params.regexp`
	Introduced in Java Agent 5.6
	Recognized from AM 7
	`org.forgerock.agents.wanted.http.url.params.regex.list`
	Introduced in Java Agent 5.6

Function	Query parameter
Type	List
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Miscellaneous (from AM 7)` Title: `Regex Query Parameters List for Policy Evaluation`

Query Parameter List for Policy Evaluation

A list of query parameters to be retained for policy evaluation and caching purposes. The property has the following format, with no spaces between values:

`[Domain[/path]]|parameter[,parameter…]`

Consider the following constraints when constructing the list of parameters:

- Add a comma (,) character at the beginning or the end of the list to retain all unnamed parameters. For example, `myapp.example.com/customers|,lang` matches both `lang` and any unnamed parameters.
- Add the asterisk (*) character to the list to retain all parameters, including unnamed ones.
- The remaining parameters (those that match the list of parameters) are sorted alphabetically.

Examples:

`org.forgerock.agents.wanted.http.url.param.list[0]=myapp.example.com/news|area`

`org.forgerock.agents.wanted.http.url.param.list[1]=example.com/news|area,country,location,`

The following incoming URL request matches a rule such as `myapp.example.com/customers|,lang`:

`http://myapp.example.com/customers?country=uk&=bristol&lang=en_GB&area=1343456`

It is cached by the agent as `http://myapp.example.com/customers?=bristol&lang=en_GB`, where both `lang` and the unnamed parameter are retained and sorted alphabetically.

Property name	`org.forgerock.agents.wanted.http.url.param.list`

Aliases	`org.forgerock.openam.agents.config.conditional.wanted.http.url.params`
	Introduced in Java Agent 5.6
	Recognized from AM 7
	`org.forgerock.agents.wanted.http.url.param.list`
	Introduced in Java Agent 5.6

Function	Query parameter
Type	List
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Miscellaneous (from AM 7)` Title: `Query Parameter List for Policy Evaluation` Legacy title: `Retain Query Parameters`

Required

AM Authentication Service Path

The path to the AM server.

Property name	`org.forgerock.agents.am.path`

Aliases	`com.iplanet.am.services.deploymentDescriptor`
	Introduced in Java Agent 5.0
	`org.forgerock.agents.am.path`
	Introduced in Java Agent 5.6

Function	Authentication service, Required
Type	String
Bootstrap property	Yes
Required property	Yes - If this property is missing, the agent fails to start
Restart required	Yes - Restart the container after changing the property

Local configuration file	`AgentBootstrap.properties`

Authentication Redirect URI

The URI the agent uses to process authentication requests.

When this property is not defined, the redirect URI is provided by AM.

When this property is defined and Location of Agent Configuration Repository is `REMOTE`, AM overwrites this property.

If OIDC authentication is being used, changing the value of this property while the agent is running prevents it from functioning. Restart the agent immediately after the value in AM is altered and the properties saved.

Property name	`org.forgerock.agents.authn.redirect.uri`

Aliases	`org.forgerock.agents.authn.redirect.uri`
	Introduced in Java Agent 5.6
	`com.sun.identity.agents.config.cdsso.redirect.uri`
	Introduced in Java Agent 5.0
	Recognized from AM 6

Function	Cross-domain single sign-on, Required
Type	String
Bootstrap property	No
Required property	No
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentConfig.properties`
AM console	Tab: `SSO` Title: `Authentication Redirect URI` Legacy title: `CDSSO Redirect URI`

Encryption Class

The name of the class used by the agent to implement encryption and decryption. The class must implement both `com.iplanet.services.util.AMEncryption` and `com.iplanet.services.util.ConfigurableKey`.

> **Warning**
>
> After changing this property, you must re-encrypt the agent profile password. For more information, refer to Encrypted Agent Password.

Property name	`org.forgerock.agents.encryptor.classname`

Aliases	`org.forgerock.agents.encryptor.classname`
	Introduced in Java Agent 5.0
	`com.iplanet.security.encryptor`
	Introduced in Java Agent 5.0

Function	Encryption, Required
Type	Classname
Default	`org.forgerock.openam.shared.security.crypto.AESWrapEncryption`
Bootstrap property	Yes
Required property	Yes - If this property is missing, the agent fails to start
Restart required	No
Local configuration file	`AgentBootstrap.properties`

AM Authentication Service Protocol

The protocol used by the AM server. Set to one of the following values:

- HTTP
- HTTPS

Property name	`org.forgerock.agents.am.protocol`

Aliases	`org.forgerock.agents.am.protocol`
	Introduced in Java Agent 5.6
	`com.iplanet.am.server.protocol`
	Introduced in Java Agent 5.0
	Recognized from AM 6

Function	Authentication service, Required
Type	String
Bootstrap property	Yes
Required property	Yes - If this property is missing, the agent fails to start
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentBootstrap.properties`

AM console	Tab: `AM Services` Title: `AM Authentication Service Protocol`

Agent Profile Realm

The realm in which the agent profile is defined.

When Enable Policy Evaluation in User Authentication Realm is `true`, AM uses this realm to evaluate polices for policy decision requests from the agent.

Property name	`org.forgerock.agents.agent.profile.realm`

Aliases	`org.forgerock.agents.agent.profile.realm`
	Introduced in Java Agent 5.6
	`com.sun.identity.agents.config.organization.name`
	Introduced in Java Agent 5.0

Function	Profile, Required
Type	String
Default	`/`
Bootstrap property	Yes
Required property	Yes - If this property is missing, the agent fails to start
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentBootstrap.properties`

Encrypted Agent Password

The agent profile password, which must correspond to the value in AM.

Set this property to the encrypted value of the password, where the password is encrypted using the key in the property Encryption Key/Salt.

Use the following command to get the encrypted value of the password, where `passwordFile` contains only the password followed by a newline, and has the access permission `400`:

```
$ ./agentadmin --encrypt agentInstance passwordFile
```

Default: Empty

Property name	`org.forgerock.agents.encrypted.password`

Aliases	`com.iplanet.am.service.secret`
	Introduced in Java Agent 5.0
	`org.forgerock.agents.encrypted.password`
	Introduced in Java Agent 5.6

Function	Miscellaneous, Required
Type	String
Bootstrap property	No
Required property	Yes - If this property is missing, the agent fails to start
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentPassword.properties`

Location of Agent Configuration Repository

The location of the agent configuration.

Property name	`org.forgerock.agents.config.location`

Aliases	`com.sun.identity.agents.config.repository.location`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.config.location`
	Introduced in Java Agent 5.6

Function	Profile, Required
Supported settings	***LOCAL*** The agent configuration is read from AgentConfiguration.properties (only). ***REMOTE*** The agent configuration is downloaded from AM.
Default	`REMOTE`
Bootstrap property	Yes
Required property	Yes - If this property is missing, the agent fails to start
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentBootstrap.properties`

AM console	Tab: `Global` Title: `Location of Agent Configuration Repository`

Agent Profile Name

The profile name used to fetch agent configuration data from AM, to evaluate policies for users, retrieve session info, and so on.

Default: Empty

Property name	`org.forgerock.agents.profile.name`

Aliases	`com.sun.identity.agents.app.username`
	Introduced in Java Agent 5.0
	`com.sun.identity.agents.config.profilename`
	Introduced in Java Agent 5.0
	`org.forgerock.agents.profile.name`
	Introduced in Java Agent 5.6

Function	Profile, Required
Type	String
Bootstrap property	Yes
Required property	Yes - If this property is missing, the agent fails to start
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentBootstrap.properties`

Autonomous mode

When `true` the agent operates independently of AM, without needing to contact an AM instance. Agents allow access to resources as defined in not-enforced lists; otherwise, they deny access.

Property name	`org.forgerock.agents.fallback.mode.enabled`

Aliases	`com.forgerock.agents.config.fallback.mode`
	Introduced in Java Agent 5.9.0
	`org.forgerock.agents.fallback.mode.enabled`
	Introduced in Java Agent 5.9.0
	`org.forgerock.agents.autonomous.mode.enabled`
	Introduced in Java Agent 5.9.0

Function	Agent, Required

Type	Boolean: `true` returns true; all other strings return `false`.
Default	`false`
Bootstrap property	Yes
Required property	Yes - If this property is missing, the agent fails to start
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentBootstrap.properties`

AM Authentication Service Host Name

The AM server host name.

Property name	`org.forgerock.agents.am.hostname`

Aliases	`com.iplanet.am.server.host`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.am.hostname`
	Introduced in Java Agent 5.6

Function	Authentication service, Required
Type	String
Bootstrap property	Yes
Required property	Yes - If this property is missing, the agent fails to start
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentBootstrap.properties`
AM console	Tab: `AM Services` Title: `AM Authentication Service Host Name`

AM Authentication Service Port

The AM server port number.

Property name	`org.forgerock.agents.am.port`

Aliases	`com.iplanet.am.server.port`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.am.port`
	Introduced in Java Agent 5.6

Function	Authentication service, Required
Type	String
Bootstrap property	Yes
Required property	Yes - If this property is missing, the agent fails to start
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentBootstrap.properties`
AM console	Tab: `AM Services` Title: `AM Authentication Service Port`

Public AM URL

The assembled "public" URL of AM. This URL is used by the agent to redirect the user's browser to AM for login (customised or not), or if necessary, exchange an SSO token for a JWT.

The following properties make up the URL:

- AM Authentication Service Protocol
- AM Authentication Service Host Name
- AM Authentication Service Port
- AM Authentication Service Path

The "private" URL is used by the agent for tasks such as establishing websockets, and obtaining authentication tokens or session information. The AM or load balancer instance can be behind a firewall to which the agent has access.

Define this property when public access to AM is restricted to a different URL from the private URL.

Property name	`org.forgerock.agents.public.am.url`

Aliases	`com.forgerock.agents.public.am.url`
	Introduced in Java Agent 5.6.3.0
	`org.forgerock.agents.public.am.url`
	Introduced in Java Agent 5.6.3.0

Function	Miscellaneous, Required

Type	String
Bootstrap property	Yes
Required property	Yes - If this property is missing, the agent fails to start
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentBootstrap.properties`

Response

Response Attribute Map

Map the value of a response attribute specified in AM to one or more HTTP headers, HTTP cookies, or request attributes, 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, HTTP cookies, or request attributes

Consider the following points for response 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 and the cookie value is different from the mapped value, the agent overwrites the cookie.
- If an HTTP cookie with the mapped name already exists and the cookie value is the same as the mapped value, the agent does nothing.
- If the profile attribute name (key) does not exist, the agent does not create the HTTP cookie.
- The agent does not automatically clear cookies. To prevent a build up of cookies, consider adding them to the Reset Cookie List.

Consider the following points for response headers:

- If an HTTP header with the mapped name does not exist, the agent creates it.
- If an HTTP header with the mapped name already exists, the agent does not overwrite it but simply appends information to the HTTP header.
- If the profile attribute name (key) does not exist, the agent does not create the HTTP header.
- When an HTTP header name is used in a request header, it is prefixed by `HTTP_`. The agent 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)`, `response attribute = COOKIE-NAME(S)`, `response attribute = REQUEST-ATTRIBUTE(S)`

Default: Empty

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`

Example: `[uid]=HEADER1|HEADER2`

Property name	`org.forgerock.agents.response.attribute.map`

Aliases	`org.forgerock.agents.response.attribute.map`
	Introduced in Java Agent 5.6
	`com.sun.identity.agents.config.response.attribute.mapping`
	Introduced in Java Agent 5.0
	Recognized from AM 6

Function	Response
Type	Map • Keys: source response attribute name • Values: one or more target response attribute names
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Application` Title: `Response Attribute Map` Legacy title: `Response Attribute Mapping`

Response Attribute Fetch Mode

Map the name of an AM policy response attribute specified in Response Attribute Map as follows:

- NONE: Do not map
- HTTP_HEADER: Map the to the name of an HTTP response header
- HTTP_COOKIE: Map to the name of an HTTP cookie
- REQUEST_ATTRIBUTE: Map to the name of an HTTP request attribute

Property name	`org.forgerock.agents.response.attribute.fetch.mode`

Aliases	`com.sun.identity.agents.config.response.attribute.fetch.mode`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.response.attribute.fetch.mode`
	Introduced in Java Agent 5.6

Function	Attributes, Response
Supported settings	***NONE*** No AM attributes are mapped. ***HTTP_HEADER*** The named AM attributes are mapped to headers in the outgoing HTTP response. ***REQUEST_ATTRIBUTE*** The named AM attributes are mapped to request attributes in the outgoing HTTP response. ***HTTP_COOKIE*** The named AM attributes are mapped to cookies in the outgoing HTTP response.
Default	`NONE`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Application` Title: `Response Attribute Fetch Mode`

SSO cookie handling

Enable SSO Token Acceptance

Set this property as follows:

- `true`: Accept SSO tokens. Use this option when the agent and the token issuer are in the same domain, and for web applications and APIs where the backend accepts user information from SSO tokens.
- `false`: Do not accept SSO tokens; require OIDC JWTs for authentication.

During session upgrade the format of the composite advice is as follows:

- When both this property and Enable Custom Login Mode are `true`, the composite advice has the following format: `?authIndexType=composite_advice&authIndexValue=<Advices Value>`
- When either property is `false`, the composite advice has the following format: `?composite_advice=<Advices Value>`

Property name	`org.forgerock.agents.accept.sso.tokens.enabled`

Aliases	`org.forgerock.agents.accept.sso.tokens.enabled`
	Introduced in Java Agent 5.7.1
	Recognized from AM 7.1
	`org.forgerock.agents.accept.sso.tokens`
	Introduced in Java Agent 5.7.1
	`com.forgerock.agents.accept.sso.tokens`
	Introduced in Java Agent 5.7.1

Function	Custom login redirect, Login redirect, SSO cookie handling
Type	Boolean: `true` returns true; all other strings return `false`.
Default	`false`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `SSO (from AM 7.1)` Title: `Enable SSO Token Acceptance` Legacy title: `Accept SSO Tokens`

Convert SSO Tokens Into OIDC JWTs

For each incoming request, the agent looks for an OIDC JWT in the cookie named by JWT Cookie Name. Set this property as follows:

- `true`: Use this value to allow users to access resources protected with systems that continue to use SSO tokens, and to use the default login redirection mode.
    - If the agent does not find a JWT in the cookie, the agent looks for an SSO token in the iPDP cookie defined during AM installation. During agent startup, the agent retrieves the name of this cookie from AM.
    - If the agent finds an SSO token in the iPDP cookie, it makes a request to AM to convert the SSO token into an OIDC JWT.

- The agent caches the SSO token, so that if it is presented in another incoming request, the agent substitutes the JWT without making a request to AM.
- If the agent does not find either token, authentication fails. The user can only access resources that are available through not-enforced rules.

- `false`: Do not convert SSO tokens into OIDC JWTs.

Property name	`org.forgerock.agents.accept.ipdp.cookie`

Aliases	`com.forgerock.agents.accept.ipdp.cookie`
	Introduced in Java Agent 5.6
	Recognized from AM 7
	`org.forgerock.agents.accept.ipdp.cookie.enabled`
	Introduced in Java Agent 5.7

Function	SSO cookie handling
Type	Boolean: `true` returns true; all other strings return `false`.
Default	`false`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `SSO (from AM 7)` Title: `Convert SSO Tokens Into OIDC JWTs` Legacy title: `Convert SSO Tokens into OpenID Connect JWTs`

SSO Cookie Domain List

When the property Enable SSO Token Acceptance is `true`, a list of domains in which the agent attempts to create SSO cookies:

- If the list is empty, the agent creates cookies only in its own domain.
- If the agent is running behind a browser, it can create cookies only in its own domain.
- If the agent is running behind a proxy, it should be able to create cookies in any required domains.

Default: Empty

Property name	`org.forgerock.agents.ipdp.cookie.domain.list`

Aliases	`org.forgerock.agents.ipdp.cookie.domain.list` Introduced in Java Agent 5.7.1 Recognized from AM 7.1
Function	SSO cookie handling
Type	List
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `SSO (from AM 7.1)` Title: `SSO Cookie Domain List`

SameSite

Set-Cookie Internal Map

When creating internal cookies, such as `am-auth-jwt` and the pre-authentication cookies, this property sets additional attributes by adding text into the `Set-Cookie` header.

Specify a key:value map, where the key is the cookie name, and the value is the string to add to the `Set-Cookie` header. If the key is omitted, the value becomes the default for all cookies.

Separate multiple values with a semicolon.

Examples:

- Set the `SameSite` attribute of the `am-auth-jwt` cookie: `org.forgerock.agents.set.cookie.internal.map[am-auth-jwt]=samesite=strict`
- Set the `SameSite` attribute of all cookies: `org.forgerock.agents.set.cookie.internal.map=samesite=strict`
- Set several attributes of `mycookie`: `org.forgerock.agents.set.cookie.internal.map[myCookie]=Max-Age=10000;Domain=.my.default.fqdn`

Property name	`org.forgerock.agents.set.cookie.internal.map`
Aliases	`org.forgerock.agents.set.cookie.internal.map` Introduced in Java Agent 5.6.3.0 Recognized from AM 7
Function	SameSite

Type	Map • Keys: Agent internal cookie name • Values: (samesite) text to be added to the Set-Cookie header
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `SSO (from AM 7)` Title: `Set-Cookie Internal Map`

Exclude Agents From Samesite Cookie Attributes

List of user agents excluded from receiving SameSite cookie attributes.

To specify different user agent patterns, add them in AM as custom properties, When user agent patterns are specified, the default list of user agents is ignored.

Property name	`org.forgerock.agents.samesite.excluded.user.agents.list`
Aliases	`org.forgerock.agents.samesite.excluded.user.agents.list` Introduced in Java Agent 5.6.3.0 Recognized from AM 7
Function	SameSite
Type	List
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `SSO (from AM 7)` Title: `Exclude Agents From Samesite Cookie Attributes` Legacy title: `Samesite Cookie Attributes Excluded User Agents Pattern List`

Set-Cookie Attribute Map

When creating cookies with the `AttributeTaskHandler`, this property sets additional attributes by adding text into the `Set-Cookie` header.

Specify a key:value map, where the key is the cookie name, and the value the string to add to the the `Set-Cookie` header.

Separate multiple values with a semicolon.

Property name	`org.forgerock.agents.set.cookie.attribute.map`
Aliases	`org.forgerock.agents.set.cookie.attribute.map` Introduced in Java Agent 5.6.3.0 Recognized from AM 7
Function	SameSite
Type	Map • Keys: Cookie name • Values: (samesite) text to be added to the Set-Cookie header
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `SSO (from AM 7)` Title: `Set-Cookie Attribute Map`

Session

Session Attribute Fetch Mode

Map the name of an AM session attribute specified in Session Attribute Map as follows:

- NONE: Do not map
- HTTP_HEADER: Map the to the name of an HTTP session header
- HTTP_COOKIE: Map to the name of an HTTP cookie
- REQUEST_ATTRIBUTE: Map to the name of an HTTP session attribute

When the value is `HTTP_COOKIE`, Cookie Reset is automatically set to `true`. Before redirecting the client for login, and when the client logs out, the agent resets profile and session attributes cookies, and cookies in the Reset Cookie List.

Property name	`org.forgerock.agents.session.attribute.fetch.mode`

Aliases	`org.forgerock.agents.session.attribute.fetch.mode`
	Introduced in Java Agent 5.6
	`com.sun.identity.agents.config.session.attribute.fetch.mode`
	Introduced in Java Agent 5.0
	Recognized from AM 6

Function	Attributes, Cookie reset, Session
Supported settings	***NONE*** No AM attributes are mapped. ***HTTP_HEADER*** The named AM attributes are mapped to headers in the outgoing HTTP response. ***REQUEST_ATTRIBUTE*** The named AM attributes are mapped to request attributes in the outgoing HTTP response. ***HTTP_COOKIE*** The named AM attributes are mapped to cookies in the outgoing HTTP response.
Default	`NONE`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Application` Title: `Session Attribute Fetch Mode`

Session Attribute Map

Map the value of an AM session attribute to one or more HTTP headers, HTTP cookies, or request attributes, depending on the value of Session Attribute Fetch Mode. The session attribute is the attribute in the session to be fetched.

- Map key: The name of an AM session attribute
- Map value: The name of one or more HTTP headers, HTTP cookies, or request attributes

Consider the following points for HTTP 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 and the cookie value is different from the mapped value, the agent overwrites the cookie.
- If an HTTP cookie with the mapped name already exists and the cookie value is the same as the mapped value, the agent does nothing.
- If the profile attribute name (key) does not exist, the agent does not create the HTTP cookie.
- The agent does not automatically clear cookies. To prevent a build up of cookies, consider adding them to the Reset Cookie List.

Consider the following points for HTTP headers:

- If an HTTP header with the mapped name does not exist, the agent creates it.
- If an HTTP header with the mapped name already exists, the agent does not overwrite it but simply appends information to the header.
- If the profile attribute name (key) does not exist, the agent does not create the response header.
- When an HTTP header name is used in a request header, it is prefixed by `HTTP_`. The agent automatically changes lower case letters to upper case, and hyphens (`-`) to underscores (`_`). For example, `CUSTOM-userid` becomes `HTTP_CUSTOM_USERID`.

Format: `session attribute = HEADER-NAME(S)`, `session attribute = COOKIE-NAME(S)`, `session attribute = REQUEST-ATTRIBUTE(S)`

Default: Empty

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`

Example: `[UserToken]=HEADER-1|HEADER-2`

Property name	`org.forgerock.agents.session.attribute.map`

Aliases	`org.forgerock.agents.session.attribute.map`
	Introduced in Java Agent 5.6
	`com.sun.identity.agents.config.session.attribute.mapping`
	Introduced in Java Agent 5.0
	Recognized from AM 6

Function	Session
Type	Map • Keys: source session attribute name • Values: one or more target session attribute names
Bootstrap property	No

Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Application` Title: `Session Attribute Map` Legacy title: `Session Attribute Mapping`

Session Cache TTL

The interval in minutes at which entries in the session cache timeout and are purged.

If an entry is not cached, the agent must retrieve session information from AM. Therefore, by default the timeout is much longer than for the policy cache.

Property name	`org.forgerock.agents.session.cache.ttl.minutes`

Aliases	`org.forgerock.agents.session.cache.ttl.minutes`
	Introduced in Java Agent 5.6
	`org.forgerock.openam.agents.config.active.session.cache.ttl.minutes`
	Introduced in Java Agent 5.0
	Recognized from AM 7

Function	Session
Type	Integer
Default	`3`
Bootstrap property	Yes
Required property	No
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentBootstrap.properties`
AM console	Tab: `Advanced (from AM 7)` Title: `Session Cache TTL`

Max Entries in Expired Session Cache

The maximum number of entries in the expired session cache. When the maximum is reached, the oldest records are overwritten.

Property name	`org.forgerock.agents.expired.session.cache.size`
Aliases	`org.forgerock.agents.expired.session.cache.size` Introduced in Java Agent 5.7.1 Recognized from AM 7.1
Function	Session
Type	Integer
Default	`500`
Bootstrap property	Yes
Required property	No
Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentBootstrap.properties`
AM console	Tab: `Advanced (from AM 7.1)` Title: `Max Entries in Expired Session Cache` Legacy title: `Expired Session Cache Max Records`

Expired Session Cache Timeout

The interval in minutes at which entries in the expired session cache timeout and are purged.

The expired session cache records sessions that have been killed by AM. Use the cache to reduce network traffic and load on AM. When the agent receives a request using an invalidated token, it rejects the request without requesting session information from AM.

Property name	`org.forgerock.agents.sso.expired.session.cache.ttl.minutes`
Aliases	`org.forgerock.agents.sso.expired.session.cache.ttl.minutes` Introduced in Java Agent 5.7.1 Recognized from AM 7.1
Function	Session
Type	Integer
Default	`20`
Bootstrap property	Yes
Required property	No

Restart required	Yes - Restart the container after changing the property
Local configuration file	`AgentBootstrap.properties`
AM console	Tab: `Advanced (from AM 7.1)` Title: `Expired Session Cache Timeout`

Timeout

Websocket Idle Timeout

The idle timeout in milliseconds for WebSockets. If the connection is not active for this time, the agent pings AM to keep the WebSocket alive.

Property name	`org.forgerock.agents.ping.websocket.after.inactive.milliseconds`
Aliases	`org.forgerock.agents.ping.websocket.after.inactive.milliseconds` Introduced in Java Agent 5.8
Function	Timeout
Type	Integer
Default	`20000`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`

Websocket Expired Timeout

The allowed ping response time in milliseconds for WebSockets. If the WebSocket does not respond to a ping within this time, the agent closes the connection.

Property name	`org.forgerock.agents.declare.websocket.dead.after.milliseconds`
Aliases	`org.forgerock.agents.declare.websocket.dead.after.milliseconds` Introduced in Java Agent 5.8
Function	Timeout

Type	Integer
Default	`40000`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`

User mapping

User Attribute Name

When the property User Mapping Mode is `HTTP_HEADER`, this property is the name of the HTTP header attribute to identify the user. The named header must be present in the incoming headers.

Property name	`org.forgerock.agents.user.mapping.mode.attribute.name`

Aliases	`com.sun.identity.agents.config.user.attribute.name`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.user.mapping.mode.attribute.name`
	Introduced in Java Agent 5.6

Function	User mapping
Type	String
Default	`employeenumber`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Global` Title: `User Attribute Name`

User Mapping Mode

Specifies where to obtain the user ID

Property name	`org.forgerock.agents.user.mapping.mode`

Aliases	`org.forgerock.agents.user.mapping.mode`
	Introduced in Java Agent 5.6
	`com.sun.identity.agents.config.user.mapping.mode`
	Introduced in Java Agent 5.0
	Recognized from AM 6

Function	User mapping
Supported settings	***USER_ID*** If "Enable User Principal Flag" (org.forgerock.agents.userid.mapping.mode.use.dn.enabled) is `true`, the user ID is set from the User Principal. Otherwise, the user ID is set from the user's session property nominated by "User Session Name" (org.forgerock.agents.userid.mapping.mode.use.session.property.name). ***PROFILE_ATTRIBUTE*** The user ID is set to the value of a named profile attribute specified by "User Attribute Name" (org.forgerock.agents.user.mapping.mode.attribute.name). ***HTTP_HEADER*** The user ID is set to the value of the HTTP header specified by the "User Attribute Name" (org.forgerock.agents.user.mapping.mode.attribute.name). ***SESSION_PROPERTY*** The user ID is set to the value of the session property specified by the "User Attribute Name" (org.forgerock.agents.user.mapping.mode.attribute.name).
Default	`USER_ID`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Global` Title: `User Mapping Mode`

User Session Name

The user is identified by the value of this property when User Mapping Mode is `USER_ID`, and Enable User Principal Flag is `false`.

Property name	`org.forgerock.agents.userid.mapping.mode.use.session.property.name`

Aliases	`com.sun.identity.agents.config.user.token`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.userid.mapping.mode.use.session.property.name`
	Introduced in Java Agent 5.6

Function	User mapping
Type	String
Default	`UserToken`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Global` Title: `User Session Name` Legacy title: `User Token Name`

Enable User Principal Flag

When the property User Mapping Mode is `USER_ID`, this flag indicates whether to identify the user through the user DN, as follows:

- If `true`, the DN is taken from universalId, retrieved from the AM user session info.
- If `false`, the user is identified by the the property User Session Name.

Property name	`org.forgerock.agents.userid.mapping.mode.use.dn.enabled`

Aliases	`com.sun.identity.agents.config.user.principal`
	Introduced in Java Agent 5.0
	Recognized from AM 6
	`org.forgerock.agents.userid.mapping.mode.use.dn.enabled`
	Introduced in Java Agent 5.6

Function	User mapping
Type	Boolean: `true` returns true; all other strings return `false`.
Default	`false`
Bootstrap property	No
Required property	No
Restart required	No
Local configuration file	`AgentConfig.properties`
AM console	Tab: `Global` Title: `Enable User Principal Flag` Legacy title: `User Principal Flag`