Installation guide

This guide describes how to install ForgeRock Access Management Java Agent.

About ForgeRock Identity Platform™ software

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

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/app`.

- Access Management 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 ForgeRock BackStage website 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. For more information, refer to 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:
   1. Unix
   2. Windows

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

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

```
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.

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

      1. Unix

      2. Windows

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

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

   b. Write the key to a file:

      1. Unix

      2. Windows

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

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

```
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 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 Identity Cloud or AM. Use this password when installing an agent.

   TIP

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:

   1. Unix

   2. Windows

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

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

   ```
   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 <u>Max Entries in Not-Enforced IP Cache</u>:

   ```
   --attributevalues
   com.sun.identity.agents.config.notenforced.ip.cache.size=2
   000
   ```

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 Identity Cloud

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

This section describes how to create a journey to authenticate Java Agent to Identity Cloud. The journey has the following requirements:

- It must be called `Agent`

- Its nodes must pass the agent credentials to the Agent Data Store Decision node.

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

1. Log in to the Identity Cloud admin UI as an administrator.

2. Click **Journeys** > **New Journey**.

3. Add a journey with the following information and click **Create journey**:
   - **Name**: `Agent`
   - **Identity Object**: The user or device to authenticate.
   - (Optional) **Description**: Authenticate an agent to Identity Cloud

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

4. Using the 🔍 **Filter nodes** bar, find and then drag the following nodes from the **Components** panel into the designer area:
   - <u>Zero Page Login Collector</u> 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.
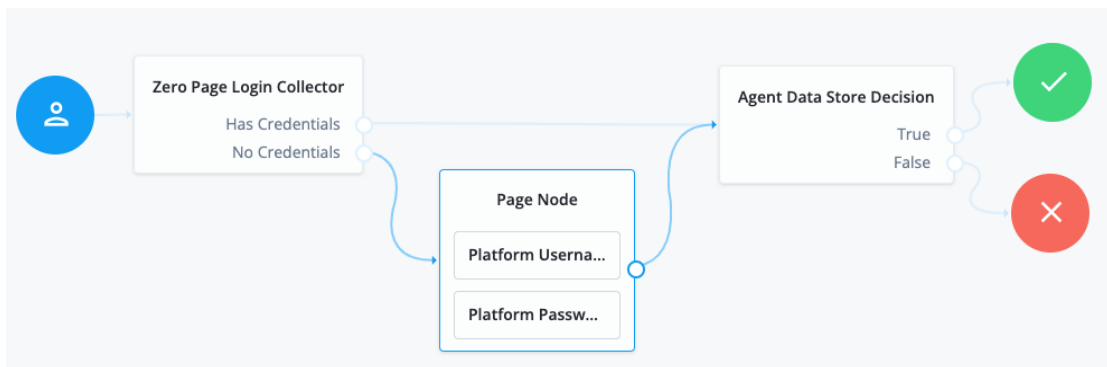
IMPORTANT

> 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 Identity Gateway, Java Agent, and Web Agent. Consider this point if you change the tree configuration.

1. On the **Realms** page of the AM admin UI, choose the realm in which to create the authentication tree.

2. On the **Realm Overview** page, click 👤 **Authentication** > **Trees** > ➕ **Create tree**.

3. Create a tree named `Agent`.

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

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

Button	Usage
⊟	Lay out and align nodes according to the order they are connected.
⤢	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:
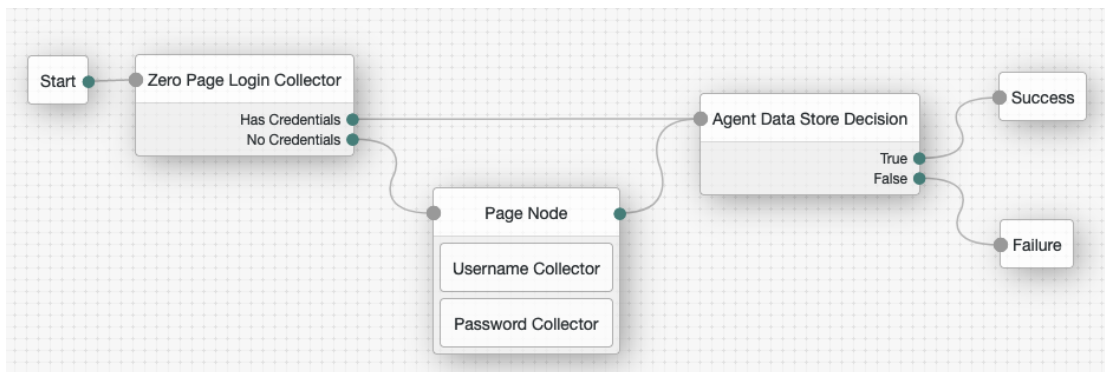
Create agent administrators for a realm

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 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 <u>Before you install</u>, and perform the steps in
   <u>Preinstallation tasks</u>.

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`.

   a. 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
      ```

   a. 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 <u>Configure an Apache HTTP Server as a</u>

reverse proxy.

b. 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
```

c. Enter the agent URL:

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

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

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

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

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

f. 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
```

g. 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/app
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 <u>Before you install</u>, and perform the steps in <u>Preinstallation tasks</u>.

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/app
```

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
```

a. 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:

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

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

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

```xml
<?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 <u>Configure an Apache HTTP Server as a reverse proxy</u>.

1. Review the information in <u>Before you install</u>, and perform the steps in <u>Preinstallation tasks</u>.

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

Command-line examples in this chapter show Jetty accessed remotely. If follow the examples and have issues accessing Jetty remotely, consider changing 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>
```

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 equivalent of 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 equivalent of 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/app
```

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
    ```

    a. 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 reference 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/startWeb
   ```

```
Logic.sh]:
/path/to/Oracle_Home/user_projects/domains/base_domain/sta
rtWebLogic.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/app
```

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
```

   a. 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/domai
    ns/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/startWebLo
gic.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.j
   ar:
    ...
   /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`: For information, refer to Configuration files.

- `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:

```xml
<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 > **Applications** > **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.

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:

a. In the AM admin UI, select **REALMS** > realm name > **Applications** > **Agents** > **Java** > agent name > **SSO**.

b. Enable <u>Transmit Cookies Securely</u>.

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/JavaVirtualMachi
  nes/jdk1.8.0_101.jdk/Contents/Home/jre/lib/security/cac
  erts
  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/c
  acerts \
  -Djavax.net.ssl.trustStorePassword=changeit"
  ```

4. Restart the agent.

Upgrade

For information about upgrade between supported versions of Java Agent, refer to
ForgeRock Product Support Lifecycle Policy | IG and Agents.

This section describes how to upgrade a single 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.

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 Release naming. For example, update from 2023.3 to 2023.6 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:

  - Do not require any update to the configuration
  - Cannot 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 Release naming. For example, upgrade from 5.10 to 2023.3 is a major
  upgrade.

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

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

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 2023.3
to the minor version 2023.6.

Tomcat Java Agent software update

1. Read the release notes for information about changes in Java Agent.

2. Download the agent binaries from the ForgeRock 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-2023.6.jar`:

   1. Unix

   2. Windows

   ```
   $ cd /opt/container
   $ find . -type f -name 'jee-agents-*.jar' -print
   ./lib/jee-agents-sdk-2023.6.jar
   ```

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

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

```
  date  time          ... jee-agents-sdk-2023.6.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-2023.6.jar`:

    1. Unix

    2. Windows

```
$ cd /opt/container
$ rm -f lib/jee-agents-sdk-2023.6.jar
$ cp /tmp/tomcat_agent/lib/jee-agents-sdk-2023.9.jar lib
```

```
C:\opt\container> del lib\jee-agents-sdk-2023.6.jar
C:\opt\container> copy C:\tmp\tomcat_agent\lib\jee-agents-sdk-
2023.9.jar lib
```

9. (Optional) Update the .jar files outside the container.

    a. Using the `.amAgentLocator` file, find the directory in which the agent was originally installed:

        1. Unix

        2. Windows

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

/path/to/java_agents/tomcat_agent
```

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

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

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

        1. Unix

        2. Windows

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

agent.jar
```

```
jee-agents-installtools-2023.6.jar
jee-agents-sdk-2023.6.jar
```

```
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-2023.6.jar
... jee-agents-sdk-2023.6.jar
```

c. Replace the files with the newer downloaded version:

  1. Unix

  2. Windows

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

agent.jar
jee-agents-installtools-2023.9.jar
jee-agents-sdk-2023.9.jar
```

```
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-2023.9.jar
C:\tmp\tomcat_agent\lib\jee-agents-sdk-2023.9.jar
```

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

  1. Unix

  2. Windows

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

```
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:

   a. 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 2023.9`.

   b. Navigate to a protected page on the website and confirm whether you can access it according to your configuration.

   c. 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 ForgeRock 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.

   a. Locate the path to the installation, for example, at
      `/path/to/jboss/modules/org/forgerock/openam/agent/main/modules/org/forgerock/openam/agent/main`.

   b. 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-2023.6.jar`

      - `agent.jar`

   c. 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-2023.6.jar"/>
      ```

      to

      ```
      <resource-root
      path="/path/to/java_agents/jboss_agent/lib/jee-agents-sdk-
      2023.9.jar"/>
      ```

7. Update the .jar files outside the container.

   a. 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
      ```

   b. 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
      ```

   c. 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:

    a. 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 2023.9`.

    b. Navigate to a protected page on the website and confirm whether you can access it according to your configuration.

    c. 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 ForgeRock 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-2023.6.jar`:

```
$ cd /path/to/java_agents/jetty_agent/lib
$ rm -f jee-agents-sdk-2023.6.jar
$ cp /tmp/jetty_agent/lib/jee-agents-sdk-2023.9.jar .
```

7. Update the Jetty configuration:

    a. Go to the Jetty base directory.

```
$ cd /path/to/jetty-base/modules
```

    b. 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.

    c. In `amlogin.mod`, update the version number for `jee-agents-sdk-version.jar`. The following example includes `jee-agents-sdk-2023.9.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-
2023.6.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:

    a. 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 2023.9` .

    b. Navigate to a protected page on the website and confirm whether you can access it according to your configuration.

    c. 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 ForgeRock 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/set AgentEnv_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.

   a. 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
   ```

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

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

   agent.jar
   jee-agents-installtools-2023.6.jar
   jee-agents-sdk-2023.6.jar
   ```

   c. 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-2023.9.jar
   jee-agents-sdk-2023.9.jar
   ```

8. Update the environment settings:

   a. Locate the `setAgentEnv_AdminServer.sh` file. The file can be in a directory such as `/Oracle/Middleware/Oracle_Home/user_projects/domains/base_domain/`.

   b. 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 .

c. 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-2023.9.jar:/path/to/java_agents/weblogic_agent/locale:/path/to/java_agents/weblogic_agent/Agent_001/config"
CLASSPATH="${CLASSPATH}${CLASSPATHSEP}${AGENT_CLASSPATH}"
export CLASSPATH
...
```

d. 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:

a. 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 2023.9` .

b. Navigate to a protected page on the website and confirm whether you can access it according to your configuration.

c. 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 a previous 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 ForgeRock 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

- 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:

    a. 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 2023.9`.

    b. Navigate to a protected page on the website and confirm whether you can access it according to your configuration.

    c. Check logs files for warnings and errors.

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

Roll back from a major upgrade

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.

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/startWeb
Logic.sh]:
/Oracle_Home/user_projects/domains/base_domain/startWebLog
ic.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
...
```

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] [--acceptLicence]`

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.

  --acceptLicense

  Do not display the license during installation.

--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] [--acceptLicence]`

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.

  --acceptLicense

  Do not display the license during installation.

--custom-install, --custom

Installs a new agent instance, specifying advanced configuration options.

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

--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.

--acceptLicense

    Do not display the license during installation.

--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.