Animal Welfare Assessment Grid (AWAG) Update Guide from v3 to v4

Contents

Update information

This guide is aimed at helping system administrators update from AWAG version 3 to 4.
The update includes:

- Functionality to manage users (database authentication mode).
- Functionality to track user actions for traceability purposes.

You will have to perform the following actions:

- Run a SQL script to update the awauth database.
- Run a SQL script to update the awdatabase database.
- Edit PostgreSQL database configuration.
- Edit Glassfish application server configuration.
- Copy new client side code to its current location on a webserver.
- Configure the contents of the war file to include changes to web.xml and glassfish.xml files.
- Deploy the new war file to Glassfish application server.

The guide assumes that you have downloaded the AWAG project release contents from the GitHub repository (https://github.com/PublicHealthEngland/animal-welfare-assessment-grid/releases).
The guide refers to location of the download as **{github-base}**.

{glassfish-base} refers to the root directory of the Glassfish install location.

{apache-install-base} refers to the root directory of Apache webserver. You may need to adjust the paths based on this according to the version of Apache/other webserver you use.

Database update

Complete the following steps in order to update the databases used by the AWAG software:

1. Use pgAdmin or move to the bin directory of your Postgres installation and run the following command:

   psql -U awag awauth < **{github-base}**/configuration/update-awauth-r3-r4.sql

2. Use pgAdmin or move to the bin directory of your Postgres installation and run the following command:

   psql -U awag awdatabase < **{github-base}**/configuration/update-awdatabase-r3-r4.sql

3. Edit the Postgres database configuration file, 'postgresql.conf'. This should be located in the Postgres installation directory: postgresql-install-root/[version]/data. For example: PostgreSQL\9.3\data

   Change the following line in 'postgresql.conf':
   #max_prepared_transactions = 0
   to:
   max_prepared_transactions = 10

4. Restart the Postgres database server.

Glassfish configuration update

Complete the following steps in order to update the Glassfish application server:

1. Edit **{glassfish-base}**/glassfish/domains/domain1/config/domain.xml
2. Find the <resources> section.
3. Change to the following line (difference in bold):
   <jdbc-connection-pool driver-classname="org.postgresql.Driver" ping="true" name="awDatabase" res-type="java.sql.Driver">
   to:
   <jdbc-connection-pool driver-classname="org.postgresql.Driver" **datasource-classname="org.postgresql.xa.PGXADataSource" res-type="javax.sql.XADataSource"** name="awDatabase" ping="true">
4. Change the following line (difference in bold):
   <jdbc-connection-pool driver-classname="org.postgresql.Driver" ping="true" name="awAuth" res-type="java.sql.Driver">
   to:
   <jdbc-connection-pool driver-classname="org.postgresql.Driver" **datasource-classname="org.postgresql.xa.PGXADataSource" res-type="javax.sql.XADataSource"** name="awAuth" ping="true">
5. **Reload the configuration by restarting the glassfish domain or server.**

Client side code update

Complete the following step in order to update the client side code installed on your webserver:

1. Copy the client side code from **{github-base}**/code/client/ into **{apache-install-base}**/www/ ensuring that you **DO NOT** overwrite **{apache-install-base}**/www/js/common/global-config.js. This file contains configuration that was tailored to your environment during the installation process therefore it does not need to be altered.

   If you overwrote the global-config.js file, please refer to the Install Notes to configure it again.

Once you have finished updating all parts of the system, you may need to refresh the client website or clear the Internet browser cache in order to see the latest changes.

War file re-configuration

As there were updates to the source code that were then re-compiled, you will need to edit the contents of the .war file again and redeploy them to glassfish:

1. Open the new .war file and copy in your existing version of index.html from the already deployed .war file as this still holds useful configuration from the first install.

Note: The .war file is just a zip file so you can use 7zip's or similar to open-archive functionality to gain access to the contents.

2. If you are using LDAP to authenticate skip to section 4. If you are using database authentication open the .war file located in **{github-base}** and ensure that the web.xml found in the WEB-INF directory contains the following:

Note: The .war file is just a zip file so you can use 7zip's or similar to open-archive functionality to gain access to the contents.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
    version="3.0">
    <display-name>aw</display-name>
    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
    </welcome-file-list>
      <!--
    <context-param>
        <param-name>authType</param-name>
        <param-value>ldap</param-value>
    </context-param>
    <login-config>
        <auth-method>FORM</auth-method>
        <realm-name>ldapRealm</realm-name>
        <form-login-config>
            <form-login-page>/login.html</form-login-page>
            <form-error-page>/login-failed.html</form-error-page>
        </form-login-config>
    </login-config>
    -->
    <context-param>
        <param-name>authType</param-name>
        <param-value>database</param-value>
    </context-param>
     <login-config>
        <auth-method>FORM</auth-method>
        <realm-name>jdbcRealm</realm-name>
        <form-login-config>
            <form-login-page>/login.html</form-login-page>
            <form-error-page>/login-failed.html</form-error-page>
```

```
            </form-login-config>
        </login-config>
        <security-constraint>
            <web-resource-collection>
                <web-resource-name>Secure Pages</web-resource-name>
                <url-pattern>/*</url-pattern>
            </web-resource-collection>
            <auth-constraint>
                <role-name>assessmentuser</role-name>
                    <role-name>admin</role-name>
            </auth-constraint>
        </security-constraint>
        <security-constraint>
            <web-resource-collection>
                <web-resource-name>Open Content</web-resource-name>
                <url-pattern>/resources/*</url-pattern>
            </web-resource-collection>
        </security-constraint>
        <session-config>
            <session-timeout>30</session-timeout>
        </session-config>
    </web-app>
```

3. Open the .war file found in **{github-base}** and ensure that glassfish-web.xml contains the following:

Note: The .war file is just a zip file so you can use 7zip's or similar to open-archive functionality to gain access to the contents.

```
<!DOCTYPE glassfish-web-app PUBLIC "-//GlassFish.org//DTD
GlassFish Application Server 3.1 Servlet 3.0//EN"
"http://glassfish.org/dtds/glassfish-web-app_3_0-1.dtd">
<glassfish-web-app>
    <!-- <security-role-mapping>
            <role-name>assessmentuser</role-name>
            <group-name>N/A </group-name>
    </security-role-mapping> -->
    <security-role-mapping>
            <role-name>admin</role-name>
            <group-name>admin</group-name>
    </security-role-mapping>
    <security-role-mapping>
            <role-name>assessmentuser</role-name>
            <group-name>assessmentuser</group-name>
    </security-role-mapping>
</glassfish-web-app>
```

4. If you are using LDAP authentication open the .war file located in **{github-base}** and ensure that the web.xml found in the WEB-INF directory contains the following:

Note: The .war file is just a zip file so you can use 7zip's or similar to open-archive functionality to gain access to the contents.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
    version="3.0">
    <display-name>aw</display-name>
    <welcome-file-list>
            <welcome-file>index.html</welcome-file>
    </welcome-file-list>
    <context-param>
            <param-name>authType</param-name>
            <param-value>ldap</param-value>
    </context-param>
    <login-config>
            <auth-method>FORM</auth-method>
            <realm-name>ldapRealm</realm-name>
            <form-login-config>
                    <form-login-page>/login.html</form-login-page>
                    <form-error-page>/login-failed.html</form-error-page>
            </form-login-config>
    </login-config>
     <!--
    <context-param>
            <param-name>authType</param-name>
            <param-value>database</param-value>
    </context-param>
     <login-config>
            <auth-method>FORM</auth-method>
            <realm-name>jdbcRealm</realm-name>
            <form-login-config>
                    <form-login-page>/login.html</form-login-page>
                    <form-error-page>/login-failed.html</form-error-page>
            </form-login-config>
    </login-config>
    -->
    <security-constraint>
            <web-resource-collection>
                    <web-resource-name>Secure Pages</web-resource-name>
                    <url-pattern>/*</url-pattern>
            </web-resource-collection>
            <auth-constraint>
                    <role-name>assessmentuser</role-name>
                        <role-name>admin</role-name>
            </auth-constraint>
```

```
    </security-constraint>
    <security-constraint>
        <web-resource-collection>
            <web-resource-name>Open Content</web-resource-name>
            <url-pattern>/resources/*</url-pattern>
        </web-resource-collection>
    </security-constraint>
    <session-config>
        <session-timeout>30</session-timeout>
    </session-config>
</web-app>
```

5. Open the .war file located in **{github-base}** and ensure that glassfish-web.xml contains the following making sure that you supply LDAP group name that your users belong to:

Note: The .war file is just a zip file so you can use 7zip's or similar to open-archive functionality to gain access to the contents.

```
<!DOCTYPE glassfish-web-app PUBLIC "-//GlassFish.org//DTD
GlassFish Application Server 3.1 Servlet 3.0//EN"
"http://glassfish.org/dtds/glassfish-web-app_3_0-1.dtd">
<glassfish-web-app>
    <security-role-mapping>
        <role-name>assessmentuser</role-name>
        <group-name>{your LDAP group to map here}</group-name>
    </security-role-mapping>
    <!--
    <security-role-mapping>
        <role-name>admin</role-name>
        <group-name>admin</group-name>
    </security-role-mapping>
    <security-role-mapping>
        <role-name>assessmentuser</role-name>
        <group-name>assessmentuser</group-name>
    </security-role-mapping>
    -->
</glassfish-web-app>
```

The LDAP group-name refers to the name of a LDAP group that contains users who will be able to log into the AWAG system. The group is mapped to 'assessmentuser' role, which is specific to the AWAG system and is used to enforce access control rules.

6. Lastly, you must redeploy the application to Glassfish. This can be done either using the auto-deploy functionality or admin GUI.