

Configuration Guide

Configuring JBoss Application Server 4.2.0 for Web Authentication with SAS 9.2 Web Applications

Configure the System for Web Authentication

This document explains how to configure Web authentication with JBoss for the SAS Web applications. Before you use this document, review “Web Authentication” in *SAS 9.2 Intelligence Platform: Security Administration Guide* to understand and verify that Web authentication is the appropriate choice for your environment.

The default security mechanism for SAS Web applications is to authenticate against the authentication provider of the SAS Metadata Server. An alternative authentication mechanism, Web authentication, is to configure JBoss to authenticate against a user registry such as an LDAP server and to configure SAS Web applications to trust the authentication that JBoss performs.

Here are the high-level steps that you must perform to configure Web authentication.

- Update the `login.config` file in your SAS configuration directory so that it contains the necessary references to the `web` domain.
- Modify the `login-config.xml` file for the JBoss server. This file contains information about the login modules that the server uses for authentication and authorization. You must modify the existing information for some login modules and add information for others.
- Add information about security constraints, an authentication method, and security roles to the SAS Logon Manager application. You must also specify the security domain that it will use.
- Configure the SAS Remote Services application so that its classpath includes the location of the JBoss classes that represent Java Authentication and Authorization Service (JAAS) principals. Logon Manager retrieves the current `Subject` from JBoss and passes it to Remote Services.
- Verify the configuration. You might need to create a `web` authentication domain and add new accounts in that domain for users.

Before beginning this procedure, shut down JBoss and Remote Services. At the end of the procedure, start Remote Services and then JBoss—in that order.

Update the login.config Configuration File

Update the *SAS-config-dir/Lev1/Web/Common/login.config* file so that the *aliasdomain* property is set to *web* in each list of properties. The file content should resemble this example:

```
PFS {
    com.sas.services.security.login.OMILoginModule    required
        "host"="metadata-server-host"
        "port"="8561"
        "repository"="Foundation"
        "domain"="DefaultAuth"
        "trusteduser"="sastrust@saspw"
        "trustedpw"="encoded-password"
        "aliasdomain"="web" /** changed from DefaultAuth to web **/
        "debug"="false";
};

SCS {
    com.sas.services.security.login.OMILoginModule    required
        "host"="metadata-server-host"
        "port"="8561"
        "repository"="Foundation"
        "domain"="DefaultAuth"
        "trusteduser"="sastrust@saspw"
        "trustedpw"="encoded-password"
        "aliasdomain"="web" /** changed from DefaultAuth to web **/
        "holdopenconnection"="true"
        "debug"="false";
};
```

Modify the Server login-config.xml File

In the *login-config.xml* file for your server, you must make some changes. You must update the SCS and PFS application policies and to add one with a name such as *SASApplicationLogin*. The latter name must match the security-domain name that you use in the *jboss-web.xml* file. The *login-config.xml* file is located in *JBoss-install-dir/server/SASServer1/conf*. (See “Making Changes to the SAS Logon Manager Application” on page 5 for information about *jboss-web.xml*.)

1. For Web authentication to work, you must edit the SCS and PFS application policies so that they resemble those in the following example:

```
<application-policy name = "PFS">
  <authentication>
    <login-module code =
      "com.sas.services.security.login.TrustedLoginModule"
      flag = "sufficient">
      <module-option name="host">metadata-server-host
    </module-option>
    <module-option name="port">8561</module-option>
    <module-option name="repository">Foundation</module-option>
    <module-option name="domain">web</module-option>
    <module-option name="aliasdomain">DefaultAuth</module-option>
    <module-option name="trusteduser">sastrust@saspw</module-option>
    <module-option name="trustedpw">encoded-password</module-option>
    <module-option name="debug">>false</module-option>
  </login-module>
```

```

<login-module code =
    "com.sas.services.security.login.TrustedLoginModule"
    flag = "optional">
    <module-option name="host">metadata-server-host
    </module-option>
    <module-option name="port">8561</module-option>
    <module-option name="repository">Foundation</module-option>
    <module-option name="domain">DefaultAuth</module-option>
    <module-option name="aliasdomain">MidtierInternal</module-option>
    <module-option name="trusteduser">sastrust@saspw</module-option>
    <module-option name="trustedpw">encoded-password</module-option>
    <module-option name="debug">false</module-option>
</login-module>
</authentication>
</application-policy>

```

```

<application-policy name = "SCS">
  <authentication>
    <login-module code =
      "com.sas.services.security.login.OMILoginModule"
      flag = "required">
      <module-option name="host">metadata-server-host
      </module-option>
      <module-option name="port">8561</module-option>
      <module-option name="repository">Foundation</module-option>
      <module-option name="domain">web</module-option>
      <module-option name="trusteduser">sastrust@saspw</module-option>
      <module-option name="trustedpw">encoded-password</module-option>
      <module-option name="debug">false</module-option>
      <module-option name="holdopenconnection">true</module-option>
    </login-module>
  </authentication>
</application-policy>

```

Replace *metadata-server-host* with the name of the machine on which your metadata server is running.

2. You must also add an application policy that indicates which login modules Logon Manager should use to authenticate users and retrieve information about roles. The outline of this section should resemble the following example:

```

<application-policy name="SASApplicationLogin">
  <authentication>
    Site-specific login module(s) goes here!
    <login-module
      code="com.sas.services.security.login.jboss.JBossTrustedLoginModule"
      flag="required">
      <module-option name="host">metadata-server-host
      </module-option>
      <module-option name="port">8561</module-option>
      <module-option name="repository">Foundation</module-option>
      <module-option name="domain">web</module-option>
      <module-option name="trusteduser">sastrust@saspw</module-option>
      <module-option name="trustedpw">encoded-password</module-option>
    </login-module>
  </authentication>
</application-policy>

```

```
</authentication>
</application-policy>
```

Above the login module shown in the example, you must supply one or more login modules that can perform these actions:

- Authenticate the user who is logging in.
- Read the security roles that the user has been assigned.

The login modules that you use depend on how you have stored your user names, passwords, and roles. However, the subsections below present examples for using text files and LDAP.

User Names, Passwords, and Roles Are Stored in Text Files

As a proof of concept, you might want to store your user names, passwords, and roles in text files and use the UsersRolesLoginModule JBoss login module. Follow these steps.

1. Add user names and passwords to this file:

```
JBoss-install-dir/server/SASServer1/conf/props/
sas-logon-mgr-users.properties
```

Add one user per line, using this format:

```
user-name=password
```

2. Add user names and roles to this file:

```
JBoss-install-dir/server/SASServer1/conf/props/
sas-logon-mgr-roles.properties
```

Here is how to assign the SASWebUser role (or whatever role name you want the application to use) to a user:

```
user-name=SASWebUser
```

The name of the role in this file must match the name of the role that you add to the Logon Manager `web.xml` file on page 6.

3. Add this XML to your `login-config.xml` file. It should be the first login module in the application policy named `SASApplicationLogin`.

```
<login-module
  code="org.jboss.security.auth.spi.UsersRolesLoginModule"
  flag="required">
  <module-option name="usersProperties">
    props/sas-logon-mgr-users.properties</module-option>
  <module-option name="rolesProperties">
    props/sas-logon-mgr-roles.properties</module-option>
  <module-option name="unauthenticatedIdentity">anonymous
  </module-option>
</login-module>
```

User Names, Passwords, and Roles Are Stored in an LDAP Directory

Here is an example that uses LDAP. It assumes that your user names, passwords, and roles are stored in an LDAP server and that part of the LDAP directory structure resembles this information:

- dc=sas, dc=com
 - ou=People
 - uid=User1
 - uid=User2
 - ou=Roles
 - Group named SASWebUser. User1 and User2 are members of this group.

You can authenticate users against this LDAP directory and read users roles by including this login module in login-config.xml.

```
<login-module
  code="org.jboss.security.auth.spi.LdapLoginModule"
  flag="required">
  <module-option name="java.naming.factory.initial">
    com.sun.jndi.ldap.LdapCtxFactory</module-option>
  <module-option name="java.naming.provider.url">
    ldap://ldap-host:port/</module-option>
  <module-option name="java.naming.security.authentication">
    simple</module-option>
  <module-option name="principalDNPrefix">uid</module-option>
  <module-option name="principalDNSuffix">
    ,ou=People,dc=sas,dc=com</module-option>
  <module-option name="rolesCtxDN">ou=Roles,dc=sas,dc=com</module-option>
  <module-option name="uidAttributeID">uniqueid</module-option>
  <module-option name="matchOnUserDN">true</module-option>
  <module-option name="roleAttributeID">cn</module-option>
  <module-option name="roleAttributeIsDN">false</module-option>
</login-module>
```

Using Other Authentication Providers

If the examples in this section do not cover your situation, consult the JBoss documentation for information about which login modules to use. In some situations you might need to use more than one module. For example, one common setup is for a company to store user names and passwords in an LDAP directory but store information about roles in a relational database. In this case, use both the LdapLoginModule and DatabaseServerLoginModule, one after another. This is referred to as stacking login modules.

Modify Logon Manager

To make the necessary changes to Logon Manager, you must edit its web.xml and jboss-web.xml files. Both files are in the WEB-INF directory. Follow these steps.

1. In a temporary directory, extract the sas.wip.apps9.2.ear (EAR) file so that you can access the Logon Manager WEB-INF directory. You can use the jar command to extract the file:

```
jar xvf sas.wip.apps9.2.ear
```

2. In a second temporary directory, extract the Logon Manager `sas.svcs.logon.war` (WAR) file:

```
jar xvf sas.svcs.logon.war
```

Within this particular directory, you can now access the Logon Manager `WEB-INF` directory.

3. Edit the `web.xml` file in the `WEB-INF` directory to add information about security constraints, an authentication method, and security roles. For example, above the `</web-app>` closing tag, you might add these elements:

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>All resources</web-resource-name>
    <url-pattern>/*</url-pattern>
    <http-method>GET</http-method>
    <http-method>POST</http-method>
  </web-resource-collection>

  <auth-constraint>
    <role-name>SASWebUser</role-name>
  </auth-constraint>
</security-constraint>

<login-config>
  <auth-method>BASIC</auth-method>
  <realm-name>myrealm</realm-name>
</login-config>

<security-role>
  <role-name>SASWebUser</role-name>
</security-role>
```

In this example, all pages are protected and can be accessed only by users who have been assigned the `SASWebUser` role.

4. Edit the `jboss-web.xml` file in the same `WEB-INF` directory. This file specifies the security domain that JBoss uses when a user tries to log in to Logon Manager. Add the line shown in bold.

```
<!-- File containing settings specific to the JBoss application server
-->
<jboss-web>
  <context-root>SASLogon</context-root>
  <security-domain>java:/jaas/SASApplicationLogin</security-domain>
</jboss-web>
```

5. Rebuild the WAR and EAR files. You can use the `jar` command to create these files:

```
jar cvf sas.svcs.logon.war *
```

```
jar cvf sas.wip.apps9.2.ear *
```

6. Copy the EAR file to the deployment directory for your server. However, do not overwrite the original EAR file unless you already made a backup copy.

Set the CLASSPATH for Remote Services JVM

Modify the classpath for Remote Services so that the Java Virtual Machine (JVM) can locate the JBoss classes in the `jbossx.jar` file that it needs when it starts. This JAR file is required and contains classes that represent JAAS principals that JVM acquires from your JBoss server.

If you are using a script to start Remote Services, you must edit the `SAS-config-dir/Lev1/Web/Applications/RemoteServices/RemoteServices.sh` file or the `RemoteServices.bat` file. To the above JAR files, add the path to this line:

```
-Dsas.app.class.path="$REMOTESERVICESDIR"
```

It should then resemble this example:

```
-Dsas.app.class.path="$REMOTESERVICESDIR:\
/install1/jboss-4.2.0.GA/server/SASServer1/lib/jbossx.jar"
```

If you are working on a Windows system and Remote Services is configured to run as a service, make this change in the `SAS-config-dir/Lev1/Web/Applications/RemoteServices/wrapper.conf` configuration file.

Start Remote Services and then JBoss.

Log On to Verify the Web Authentication Configuration

If your site was migrated from a previous SAS release—namely, it has user IDs and authentication domains that are already registered in metadata—try logging on to a SAS Web application such as SAS Web Report Studio. Otherwise, follow these steps to test and confirm that Web authentication is properly configured.

1. Use SAS Management Console to create an authentication domain named `web`.
 - a. Right-click **User Manager** and select **Authentication Domains**.
 - b. Click **New**, enter `web` in the **Name** field, and click **OK**.
2. Choose a trial user ID that exists in your user registry, such as LDAP. Use SAS Management Console to create a user definition for the user in the `web` authentication domain. Do not enter a password for the account.
3. Try logging on to a SAS Web application with the user ID.

If the log-on attempt fails, view the SAS Metadata Server log. Look for the format of the user ID that the log-on attempt used. Use SAS Management Console to modify the user definition so that the user account in the `web` authentication domain matches the user ID in the log. While you are troubleshooting, do not enter a password in the user definition because it has no effect on Web authentication. Also, do not try logging on with an internal account such as `sasadm@saspw`.

Note: As part of Web authentication, the user ID but not the password is checked against the user accounts that are stored in the SAS Metadata Repository. The user ID used to authenticate with the user registry must match exactly the user ID string found on the SAS Metadata Server for authentication to succeed. For example, if `joe` is the user ID in your user registry, the exact user ID string "`joe`" must also be found in the SAS Metadata Repository without a prefixed domain name.

Recommended Reading

SAS Institute, Inc., 2009. *SAS 9.2 Intelligence Platform: Security Administration Guide*. Cary, NC: SAS Institute, Inc. Available at <http://support.sas.com/92administration>.

SAS and all other SAS Institute product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. Other brand and product names are registered trademarks or trademarks of their respective companies.

® indicates USA registration.

Copyright © 2008 SAS Institute Inc., Cary, NC, USA. All rights reserved.