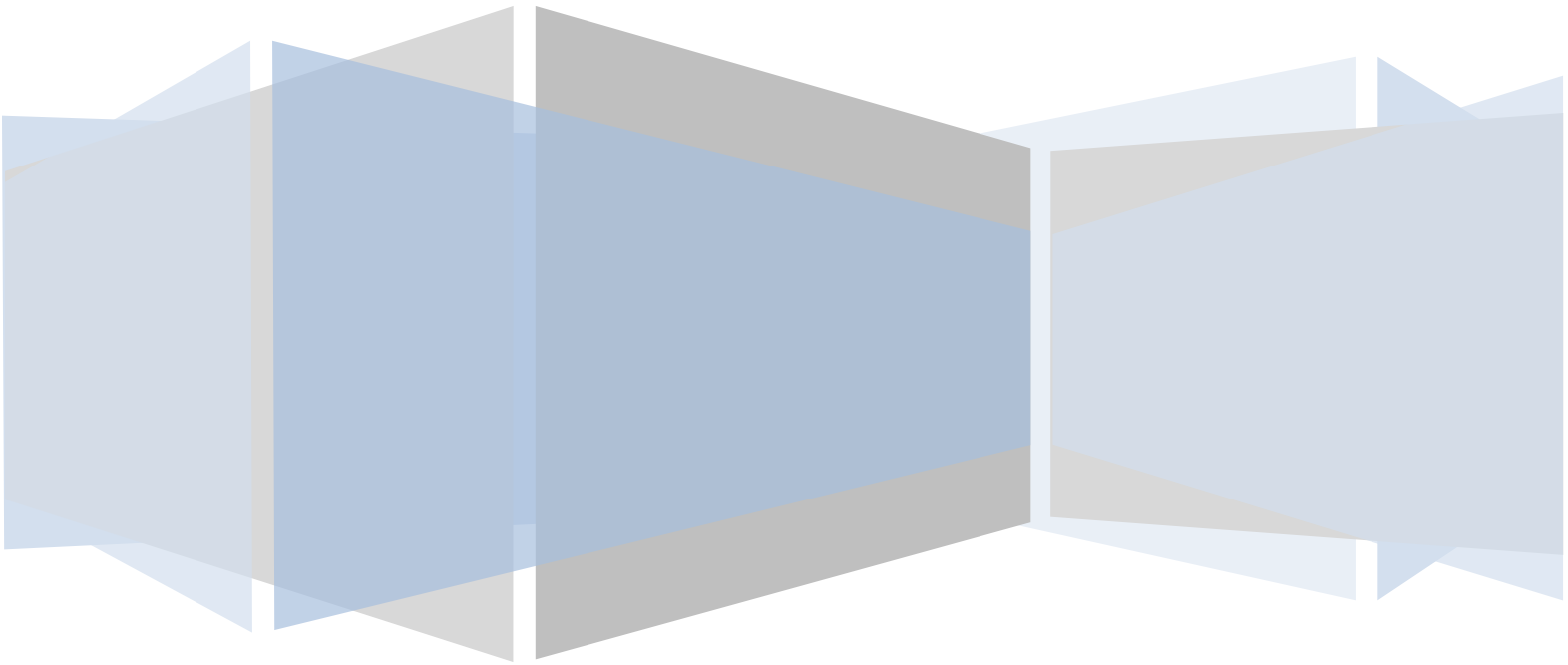
SSO Plugin

Authentication service for HP, Kinetic, Jasper, SAP and CA products

J System Solutions

http://www.javasystemsolutions.com

Version 3.6



Introduction

This guide provides SSO implements to a range of products using the SSO Plugin Authentication Service. This is a standalone edition of SSO Plugin that can be added to third party applications and in conjunction with a little configuration, brings all of the SSO Plugin integrations (Active Directory, AD, X509, etc) to the third party application.

The JSS support website contains all the SSO Plugin documentation and videos covering installation and configuration.

Implementing SSO

There are a number of steps to this process and they are split into three categories:

1. Copying files to the target web application (see below).
2. Configuring the target web application by modifying files such as the application's web.xml file.
3. Configuring SSO Plugin, of which more information can be found in the Configuring SSO Plugin document (called configuring-midtier-webtier.pdf) - this is currently bias towards BMC Mid Tier and HP Web Tier, but the SSO configuration instructions relevant.

Prior to performing the steps, stop the Java web server (ie Tomcat) instance running the application. When the steps have been completed, start it again.

Copying the SSO Plugin files to the target application

Locate the authentication-service directory within the SSO Plugin Authentication Service download package.

Copy the contents to the target web application, ie tomcat/webapps/application, where application is src, Asset Manager, jasperserver, etc.

No files will be replaced, only added to the application.

Next, proceed to configure the application following the instructions in the relevant section below.

Configuring HP Service Request Catalog 1.3+

Start by copying the SSO Plugin files to the target web application.

Deployment approaches

We provide two approaches for deploying SSO Plugin to SRC. This approach is suitable if you do not require user aliasing and do not wish to use built-in Active Directory authentication, which due to a bug in the Adobe Flash browser plugin, causes the product to fail when Internet Explorer negotiates with NTLM.

In the event your requirements include user aliasing or AD integration, consult the installation for Service Request Catalog document for the alternative strategy.

Patching application files

The following files require modifying:

web.xml

The authentication-service/WEB-INF directory contains a file called web.xml.patch.src. Open the file and copy the contents to the clipboard.

Locate the SRC web.xml file, located in tomcat/webapps/src/WEB-INF/web.xml. Make a backup of the file.

Open it in a text editor and locate the following before adding the patch (which is highlighted in bold):

```
<context-param>
  <param-name>contextClass</param-name>
  <param-value>
   com.hp.service.catalog.server.web.context.CustomXmlWebApplicationContext
  </param-value>
</context-param>
<context-param>
  <param-name>jss.backend</param-name>
  ...
```

applicationContext.properties

Locate the applicationContext.properties file, found in the SRC program directory, ie tomcat/webapps/src/WEB-INF/classes/applicationContext.properties. The required changes to enable SSO are different for SRC versions 1.3 and 1.4.

SRC 1.3

Locate the following:

```
# Security Mode: Choose your security method
```

And uncomment to enable the following option, commenting whatever is currently enabled:

```
src.security.mode=remoteUsrSsoUiAndTsoWs
```

SRC 1.4

Locate the src.security.mode variable and set to tso:

```
src.security.mode=tso
```

Next, locate the src.security.ssoEnabled variable and set it to true:

```
src.security.ssoEnabled=true
```

Modifying the Spring security configuration

The Spring security file is located in the SRC web application WEB-INF/spring/security directory. The filename is different in SRC 1.3 and 1.4 but the same change is made for both versions.

Locate the file for the relevant version:

- SRC1.3: applicationContext-security-remoteUsrSsoUiAndTsoWs.xml.
- SRC1.4: applicationContext-security.xml.

Open the file in a text editor, find the section below and add the text in bold:

```
<http entry-point-ref="authenticationProcessingFilterEntryPoint">
  <intercept-url pattern="/jss-sso/**" filters="none" />
  <intercept-url pattern="/logout.jsp" filters="none" />
  <intercept-url pattern="/secure/nosso.jsp" filters="none" />
```

Enabling manual (no SSO) login

SRC combines the login page with the main application, providing no manual (no SSO) login functionality. If you wish to enable manual login, allowing a user to see a login page when they click logout, follow these steps:

1. Rename the logout.jsp to logout.jsp.old.
2. Using your favourite text editor, create a new logout.jsp with the following content:

```
<% response.sendRedirect(request.getContextPath()+"/secure/nosso.jsp"); %>
```

3. Copy the src/secure/main.jsp to src/secure/nosso.jsp.
4. Rename the index.jsp to index.jsp.old.
5. Using your favourite text editor, create a new index.jsp with the following content:

```
<% response.sendRedirect(request.getContextPath()+"/secure/main.jsp"); %>
```

Accessing the application

When installation is complete, restart the Java web server and go to the SSO Plugin configuration interface to configure SSO: http://host/src/jss-sso/index.jsp

The default password for the configuration interface is jss.

After configuring SSO, use the Test SSO page to verify the username matches the usernames in the application and navigate the application homepage (http://host/src).

Configuring HP Asset Manager

Start by copying the SSO Plugin files to the target web application.

Patching application files

The following files require modifying:

web.xml

The authentication-service/WEB-INF directory contains a file called web.xml.patch.am. Open the file and copy the contents to the clipboard.

Locate the SRC web.xml file, located in tomcat/webapps/src/WEB-INF/web.xml. Make a backup of the file.

Open it in a text editor and locate the following before adding the patch (which is highlighted in bold):

```
<context-param>
  <param-name>contextClass</param-name>
  <param-value>

com.hp.service.catalog.server.web.context.CustomXmlWebApplicationContext
  </param-value>
</context-param>
<context-param>
  <param-name>jss.backend</param-name>
  ...
```

decorators.xml

Locate the decorators.xml file, found in the AM program directory, ie tomcat/webapps/Asset Manager/WEB-INF/cwc.

Add the following text in bold and save the file:

```
<excludes>
  <pattern>/jss-sso/**</pattern>
```

application-context.xml

Locate the application-context.xml file, found in the AM program directory, ie tomcat/webapps/Asset Manager/WEB-INF/classes/application-context.xml.

Open the file in a text editor, find the section below and add the text in bold:

```
   <property
name="convertUrlToLowerCaseBeforeComparison"><value>false</value></property
>
   <property name="publicResources">
     <list>
       <value>/jss-sso/**</value>
```

Accessing the application

When installation is complete, restart the Java web server and go to the SSO Plugin configuration interface to configure SSO: http://host/Asset_Manager/jss-sso/index.jsp

The default password for the configuration interface is jss.

After configuring SSO, use the Test SSO page to verify the username matches the usernames in the application and navigate the application homepage (http://host/Asset_Manager/index.jsp).

Configuring Kinetic Request

Kinetic Request 5.0.x is not compatible with SSO Plugin 3.6+. Please use SSO Plugin 3.5.x with KR 5.0.x, and SSO Plugin 3.6+ with KR 5.1+. You can find more information on the Kinetic Request SSO Plugin adapter on their website.

Start by copying the SSO Plugin files to the target web application. When configuring the SSO Plugin Authentication Service, enter the value kinetic.request.manual.login into the "Manual login key" field.

Patching application files

The following files require modifying:

web.xml

The authentication-service/WEB-INF directory contains a file called web.xml.patch.kr. Open the file and copy the contents to the clipboard.

Locate the Kinetic Request (KR) web.xml file, located in tomcat/webapps/kinetic/WEB-INF/web.xml. Make a backup of the file.

Open it in a text editor and locate the following before adding the patch (which is highlighted in bold):

```
<display-name>Kinetic Data Survey/Request</display-name>
<description>Deliver online surveys ...</description>
<context-param>
  <param-name>jss.backend</param-name>
  …
```

Installing the SSO adapter from Kinetic

This information is correct at the time of writing, but the Kinetic package provides more detailed instructions. Kinetic will supply two files for this integration: JSSAuthenticator.jar and JSSAuthenticator.properties.

1. Copy the JSSAuthenticator.jar file into the Kinetic WEB-INF/lib directory.
2. Copy the JSSAuthenticator.properties file into the Kinetic WEB-INF/classes directory.

The package provided by Kinetic contains detailed instructions on the various property values and configuration information, which is also summarised in this document.

Configuring Kinetic Request

Navigate to: http://host/kinetic/AdminConsole and login as an administrator. Make the following changes:

1. Under properties, enter com.kd.kineticSurvey.authentication.JSSAuthenticator into the SSO Adapter class field.
2. Under properties, enter the full pathname to the JSSAuthenticator.properties file in the SSO Adapter properties field.

Restart Tomcat.

Accessing the application

When installation is complete, restart the Java web server and go to the SSO Plugin configuration interface to configure SSO: http://host/kinetic/jss-sso/index.jsp

The default password for the configuration interface is jss.

After configuring SSO, use the Test SSO page to verify the username matches the usernames in the application.

To make a survey accessible with SSO, select "Allow anonymous" in the Advanced tab, and under the Audit tab, select requires authentication and external in the drop down selector.

Finally, navigate to the survey and it should be accessible with SSO.

Configuring Kinetic Calendar

Start by copying the SSO Plugin files to the target web application.

Patching application files

The following files require modifying:

web.xml

The authentication-service/WEB-INF directory contains a file called web.xml.patch.kc. Open the file and copy the contents to the clipboard.

Locate the Kinetic Calendar (KC) web.xml file, located in tomcat/webapps/KinCal/WEB-INF/web.xml. Make a backup of the file.

Open it in a text editor and locate the following before adding the patch (which is highlighted in bold):

```
<display-name>Kinetic Calendar</display-name>
<description>Actionable online calendar for BMC Remedy.</description>
<context-param>
  <param-name>jss.backend</param-name>
  …
```

KinCal.xml

Locate the KinCal.xml file, which is located in the tomcat/webapps/KinCal/WEB-INF/classes directory.

Add the following text in bold:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
  <entry key="AuthenticatePublicPage">true</entry>
```

Restart Tomcat.

Accessing the application

When installation is complete, restart the Java web server and go to the SSO Plugin configuration interface to configure SSO: http://host/KinCal/jss-sso/index.jsp

The default password for the configuration interface is jss.

After configuring SSO, use the Test SSO page to verify the username matches the usernames in the application.

Finally, navigate to Kinetic Calendar and it should be accessible with SSO.

Configuring Jasper Reports 4.1/4.5/5

This integration has no dependency on BMC or HP ITSM. A separate integration document demonstrates how to configure Jasper Reports when integration with these products is required, allowing users and groups to automatically be synchronised between ITSM and Jasper Reports.

Start by copying the SSO Plugin files to the target web application.

Patching application files

The following files require modifying:

web.xml

The authentication-service/WEB-INF directory contains a file called web.xml.patch.jasper. Open the file and copy the contents to the clipboard.

Locate the Jasper Server web.xml file, located in jasperreports/apache-tomcat/webapps/jasperserver/WEB-INF/web.xml. Make a backup of the file.

Open it in a text editor and locate the last context-param before adding the patch (which is highlighted in bold):

```
<context-param>
  <param-name>webAppRootKey</param-name>
  <param-value>jasperserver.root</param-value>
</context-param>
<context-param>
  <param-name>jss.backend</param-name>
  …
```

applicationContext-security.xml

Locate the file jasperreports/apache-tomcat/webapps/jasperserver/WEB-INF/applicationContext-security.xml.

Open the file in a text editor, find the section below and add the text in bold:

```
<beans …>
  <bean id="jss.j2eefilter"
class="com.javasystemsolutions.integrations.spring.security.SSOPluginPreAuthFilter">
    <property name="authenticationManager"><ref
local="authenticationManager"/></property>
  </bean>
  <bean id="jss.preAuthenticatedAuthenticationProvider"
class="org.springframework.security.providers.preauth.PreAuthenticatedAuthenticationProvider">
    <property name="preAuthenticatedUserDetailsService">
      <bean
class="com.javasystemsolutions.integrations.jasper.SSOPluginUserDetailService">
        <property name="sessionFactory" ref="sessionFactory" />
        <property name="userAuthorityService"
ref="jss.txproxy.userAuthorityService" />
        <property name="tenantService" ref="jss.txproxy.tenantService" />
        <property name="tenantMap"><map>
```

```
          <entry key="localdomain.local" value="Test" />
        </map></property>
      </bean>
    </property>
  </bean>
  <bean id="jss.txproxy.tenantService"
class="org.springframework.transaction.interceptor.TransactionProxyFactoryB
ean">
    <property name="transactionManager" ref="transactionManager"/>
    <property name="target" ref="${bean.hibernateTenantService}"/>
    <property name="transactionAttributes"><props>
      <prop key="get*">PROPAGATION_REQUIRED</prop>
      <prop key="*">PROPAGATION_SUPPORTS</prop>
    </props></property>
  </bean>
  <bean id="jss.txproxy.userAuthorityService"
class="org.springframework.transaction.interceptor.TransactionProxyFactoryB
ean">
    <property name="transactionManager" ref="transactionManager"/>
    <property name="target" ref="${bean.internalUserAuthorityService}"/>
    <property name="transactionAttributes"><props>
      <prop key="get*">PROPAGATION_REQUIRED</prop>
      <prop key="put*">PROPAGATION_REQUIRES_NEW</prop>
      <prop key="update*">PROPAGATION_REQUIRES_NEW</prop>
      <prop key="*">PROPAGATION_SUPPORTS</prop>
    </props></property>
  </bean>
```

Find the following, and add the text in bold:

```
  <bean id="authenticationManager"
class="org.springframework.security.providers.ProviderManager">
    <property name="providers">
      <list>
        <ref bean="jss.preAuthenticatedAuthenticationProvider"/>
```

applicationContext-security-web.xml

Locate the file jasperreports/apache-tomcat/webapps/jasperserver/WEB-INF/applicationContext-security-web.xml.

Open the file in a text editor, find the section below and add the text in bold:

```
  <bean id="filterChainProxy"
class="org.springframework.security.util.FilterChainProxy">
    <property name="filterInvocationDefinitionSource">
      <value>
        CONVERT_URL_TO_LOWERCASE_BEFORE_COMPARISON
        PATTERN_TYPE_APACHE_ANT
        ...
/**=httpSessionContextIntegrationFilter,jss.j2eefilter,multipartRequestWrap
perFilter,...
      </value>
    </property>
  </bean>
```

In the same area, remove references to JIAuthenticationSynchronizer and a comma. It needs to be removed from each of the lines under <value>..</value>, ie. remove the text in bold:

```
/**=httpSessionContextIntegrationFilter,...JIAuthenticationSynchronizer,...
```

decorators.xml

Locate the file jasperreports/apache-tomcat/webapps/jasperserver/WEB-INF/decorators.xml.

Open the file in a text editor, find the section below and add the text in bold:

```
<excludes>
  <pattern>/jss-sso/*</pattern>
```

Mapping Windows domains to tenants

For organisations using the Jasper Reports enterprise edition, Windows domains can be mapped to tenants, mapping newly created users from a domain to a specific tenant. The tenants must exist and the following configuration (added above) is used to provide this mapping - each entry line maps a Windows DNS domain (the key) to the tenant (the value).

```
<property name="tenantMap"><map>
  <entry key="uk.corporate.com" value="UK" />
  <entry key="us.corporate.com" value="US" />
</map></property>
```

Accessing the application

When installation is complete, restart the Java web server and go to the SSO Plugin configuration interface to configure SSO: http://host/jasperserver/jss-sso/index.jsp

The default password for the configuration interface is jss.

After configuring SSO, use the Test SSO page to verify the username matches the usernames in the application.

Finally, navigate to Jasper Server with the following URL and SSO should take place: http://host:9080/jasperserver/

You can go directly to the login page by visiting this URL: http://host:9080/jasperserver/login.html

Start by copying the SSO Plugin files to the target web application.

Configuring SAP Business Objects

Start by copying the SSO Plugin files to the target web application.

The following steps outline the integration process.

1. The authentication-service/WEB-INF directory contains a file called web.xml.patch.boxi. Open the file and copy the contents to the clipboard.
2. Locate the Business Objects InfoViewApp web.xml file, usually located in C:\Program Files\Business Objects\Tomcat55\webapps\InfoVInfoViewApp\WEB-INF\web.xml. Make a backup of the file.
3. Open the InfoViewApp web.xml file in a text editor and locate the last <context-param> element. Immediately after it, paste the patch copied to the clipboard in step 1, which is highlighted in bold for illustrative purposes:

```
<context-param>
   <param-name>path.rightFrame</param-name>
   <param-value>1</param-value>
</context-param>
<context-param>
  <param-name>jss.backend</param-name>
  …
```

4. Locate the following section of the web.xml file:

```
<context-param>
  <param-name>trusted.auth.user.retrieval</param-name>
  <param-value></param-value>
</context-param>
```

and set the value to USER_PRINCIPAL, ie.

```
  <param-value>USER_PRINCIPAL</param-value>
```

5. Locate the dswsbobje directory, typically found in C:\Program Files\Business Objects\Tomcat55\webapps\. Locate the axis2.xml in the WEB-INF\conf directory, open it in a text editor and search for the following, changing true to false (highlighted in bold):

```
<parameter name="disableREST" locked="true">false</parameter>
```

6. Create a file in the Business Objects installation directory (C:\Program Files\Business Objects\BusinessObjects Enterprise 12.0\win32_x86) called TrustedPrincipal.conf with the following content:

```
SharedSecret=somerandomvalue
```

7. Login to the CMC and go to the Authentication, Enterprise, and tick 'Trusted Authentication is enabled'. Enter the value set in (**somerandomvalue** in our example) key above into the 'Shared secret' field. Press update.
8. Restart the Business Objects Tomcat instance.

Accessing the application

When installation is complete, go to the SSO Plugin configuration interface to configure SSO: http://host/InfoViewApp/jss-sso/index.jsp. The default password for the configuration interface is jss.

After using the SSO Plugin Test SSO function to ensure you have SSO access, and ensuring an account exists in BOXI with the same username, navigate to http://host/InfoViewApp/logon/logon.do and you should be logged in with SSO.

Configuring CA Clarity PPM

Start by copying the SSO Plugin files to the target web application.

The following steps outline the integration process.

1. The authentication-service/WEB-INF directory contains a file called web.xml.patch.generic. Open the file and copy the contents to the clipboard.
2. Locate the CA Clarity PPM web.xml file, usually located in niku/webroot/WEB-INF/web.xml. Make a backup of the file.
3. Open the Clarity web.xml file in a text editor and locate the <display-name> element. Immediately after it, paste the patch copied to the clipboard in step 1, which is highlighted in bold for illustrative purposes:

```
<display-name>Clarity Web</display-name>
<context-param>
  <param-name>jss.backend</param-name>
  …
```

4. Restart the Clarity server.
5. Navigate to the Clarity administration screen and locate the single-sign-on section.
    a. Locate the Token Name field and enter jss.sso.username
    b. Locate the Token Type selector and select Header.
    c. Locate and check the Use Single Sign-On checkbox.
6. Restart the Clarity server.

Accessing the application

When installation is complete, go to the SSO Plugin configuration interface to configure SSO: http://host/niku/jss-sso/index.jsp. The default password for the configuration interface is jss.

After using the SSO Plugin Test SSO function to ensure you have SSO access, and ensuring an account exists in Clarity with the same username, navigate to the default Clarity URL (http://host/niku) and you should be logged in with SSO.

Configuring Jive Software 4.5+

Start by copying the SSO Plugin files to the target web application, keeping in mind that the Jive application root will be a path similar to /usr/local/jive/applications/template/application.

Unfortunately, Jive blocks the SSO Plugin configuration interface so the authentication-service.war file must be deployed to a Tomcat instance in order to configure the relevant SSO integration. The jss-ssoplugin.properties file can then be copied from the authentication service web application to the patched Jive application, typically placed in /usr/local/jive/applications/template/application/WEB-INF/classes.

The following steps outline the integration process - it is assumed [JIVE_HOME] points to the jive installation, typically /usr/local/jive.

1. The authentication-service/WEB-INF directory contains a file called web.xml.patch.generic. Open the file and copy the contents to the clipboard.
2. Locate the Jive web.xml file, usually located in [JIVE_HOME] /applications/template/application/WEB-INF/web.xml. Make a backup of the file.
3. Locate the jss-sso-jive.jar file in the jive-software directory and copy it to the [JIVE_HOME] / applications/template/application/WEB-INF/lib directory.
4. Open the Jive web.xml file in a text editor and locate the last </filter> element. Immediately after it, paste the patch copied to the clipboard in step 1, which is highlighted in bold for illustrative purposes:

```
<filter>
   <filter-name>hostFilter</filter-name>
   <filter-class>org.apache.shindig.common.servlet.HostFilter</filter-
class>
</filter>
<context-param>
  <param-name>jss.backend</param-name>
  …
```

5. In the web.xml, locate and remove the following:

```
<listener>
  <listener-
class>com.jivesoftware.community.aaa.JiveHttpSessionAttributeListener</list
ener-class>
</listener>
```

6. Create a directory [JIVE_HOME]/jss-sso.
7. Locate the jive-spring-jss.xml file in the jive-software directory and copy it to the [JIVE_HOME] /jss-sso directory.
8. Edit the [JIVE_HOME] /applications/sbs/bin/setenv file and locate the line:

```
# Place holder for custom application options
```

   Place the following line after the line above:

```
CUSTOM_OPTS="-Djive.extensionPath=[JIVE_HOME]/jss-sso"
```

9. Restart the Jive server.

Configuring username integration

By default, the SSO username as configured through the Authentication Service configuration interface and reported through the Test SSO page, is used to query the Jive database for a matching user account.

For organisations with more complicated Jive usernames, such as user@dns.domain, a username format can be created using the user aliasing variables defined in the Configuring SSO Plugin document. The username format is configured in the jive-spring-jss.xml file:

```
<bean id="ssoPluginAuthenticationFilter"
class="com.javasystemsolutions.sso.integrations.jive.JiveSpringSecurityFilt
er" init-method="setup">
  ...
  <property name="userFormat" value="$SSO_USERNAME$@$SSO_DNS_DOMAIN$" />
```

Accessing the application

Jive does not provide an easy way to separate SSO and non-SSO, ie a single point of entry to access Jive using SSO. Therefore, the entire application is protected with SSO Plugin. If no matching Jive user is found in the Jive database, the user is presented with a login page.