SSO Plugin

Authentication Service for HP, Kinetic and Jasper products

J System Solutions

http://www.javasystemsolutions.com

Version 3.5

Table of Contents

Introduction

This guide provides an SSO implementation for HP, Kinetic and Jasper products.

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 web.xml.
3. Configuring SSO Plugin, of which more information can be found in the Configuring SSO Plugin document - 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 files to the target application

Locate the authentication-service directory in the installation files. 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.

Configuring HP Service Request Catalog

HP Service Request Catalog

There is another approach to providing SSO for SRC which should (assuming HP resolve an issue with the product) be suitable for SRC version 1.4. If you have used 'user aliasing' when implementing SSO for Service Manager Web Tier, the alternative approach must be followed for all versions of SRC. Otherwise, this method is suitable.

If in any doubt, discuss your requirements with JSS.

Known issues with SRC

There are a few known issues when implementing SSO for SRC and these have been raised with HP's SRC product development team. The relevant issues are detailed below.

Manual login

The manual login page does not work when SSO is enabled. HP have confirmed this to be an issue and have informed us it will be fixed in SRC version 1.4.

A solution has been provided by JSS in the instructions below.

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.CustomXmlWebApplicatio
nContext
  </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.

This file is the SRC configuration file that is modified to specify the location of Service Manager. It also specifies the authentication method. Locate the following:

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

and select:

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

applicationContext-security-remoteUsrSsoUiAndTsoWs.xml

Locate the applicationContext-security-remoteUsrSsoUiAndTsoWs.xml file, found in the SRC program directory. ie tomcat/webapps/src/WEB-INF/spring/security/applicationContext-security-remoteUsrSsoUiAndTsoWs.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 the manual login page

Unfortunately, the SRC application combines a login page with the main application, so it is difficult to separate SSO and non-SSO access given the main application is protected via SSO.

To work around this problem, the following steps will result in the logout process presenting a login page:

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.

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

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.CustomXmlWebApplicatio
nContext
  </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></p
roperty>
   <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

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

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.

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 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

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

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="org.springframework.security.ui.preauth.j2ee.J2eePreAuthentic
atedProcessingFilter">
        <property name="authenticationManager"><ref
local="authenticationManager"/></property>
    </bean>
    <bean id="jss.preAuthenticatedAuthenticationProvider"
class="org.springframework.security.providers.preauth.PreAuthenticat
edAuthenticationProvider">
      <property name="preAuthenticatedUserDetailsService">
            <bean
class="com.javasystemsolutions.integrations.spring.security.SSOPlugi
nUserDetailService">
              <property name="sessionFactory" ref="sessionFactory"
/>
```

```
              <property name="userAuthorityService"
ref="jss.txproxy.userAuthorityService" />
            </bean>
      </property>
    </bean>
    <bean id="jss.txproxy.userAuthorityService"
class="org.springframework.transaction.interceptor.TransactionProxyF
actoryBean">
      <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_REQUIRED</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.j2eefilt
er,multipartRequestWrapperFilter,...
            </value>
        </property>
    </bean>
```

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