

How to Set Up Single Sign-On Between an IBM WebSphere Portal and the SAP Enterprise Portal Using JAAS

SAP ENTERPRISE PORTAL 6.0 SP 10

The texts, references and graphics contained in this manual have been compiled with utmost care; nevertheless, it is impossible to guarantee that they are fully without error. IBM and SAP cannot assume any responsibility for the correctness or completeness of the following documentation; the user alone is responsible for verifying the information contained therein.

IBM and SAP will only assume liability for damage arising from the use of this documentation – irrespective of the pertinent legal basis – in the case of intentional or active negligence; under no other circumstances will a warranty be made.



- **This version: March 2005**
- **Authors:**
 - **Patrick Höfer, btexx business technologies**
 - **Felix Huber, SAP Deutschland AG & Co. KG**
 - **Thomas Csapo, SAP Deutschland AG & Co. KG**

1 Table of Contents

1	Table of Contents	2
2	Scenario	3
3	Solution	3
4	Details of Solution	4
4.1	Developing and Deploying a WSP Portlet	4
4.2	Setting Up the SSL Keys and Certificates on the WSP (Optional)	2
4.3	Setting Up the SSL Keys and Certificates on SAP EP (Optional)	2
4.4	Developing and Deploying the SAP EP JAAS Module	2
4.5	Configure the SAP EP JAAS Module	12
4.6	Debug the SAP EP JAAS Module	16
4.7	Configuring SSO and Session Parameter in WSP Administration Console	16

2 Scenario

This guide provides instructions for setting up a single sign-on (SSO) scenario using an IBM WebSphere Portal (WSP) as the leading portal accessing an SAP Enterprise Portal (SAP EP). After launching SAP EP from WSP, you can log on to SAP EP with the same user that is already logged on to the WSP.

There is only one limitation to consider: Both WSP and SAP EP must either use the same user repository or have a user with the same ID defined for both portals, independent of their repositories.

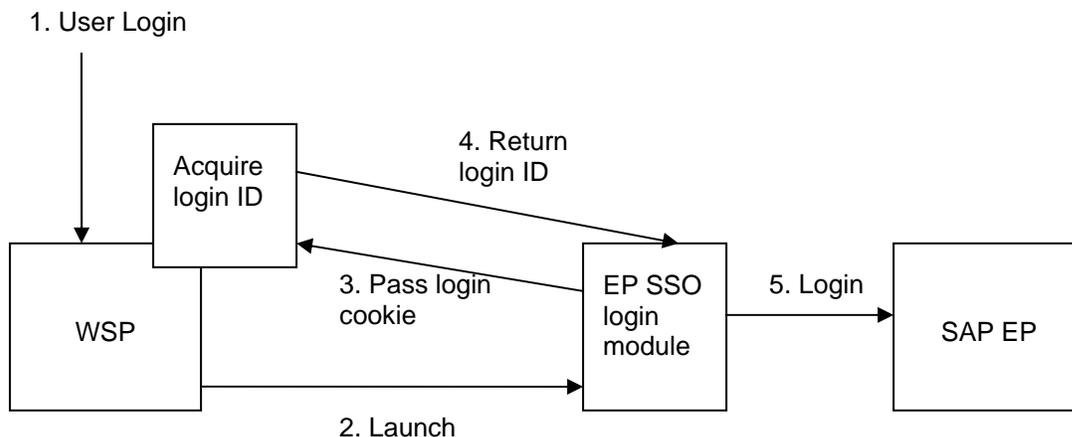
3 Solution

When a request is sent to the SAP EP login page (`http://<hostname>:<EP port#>/irj/portal`), a login module is activated that determines whether the current user is authenticated and has access to the SAP EP contents. The active login module is determined according to the SAP EP configuration.

In this solution for SSO, a new login module needs to be developed. This new login module gets the user ID from WSP and logs on to SAP EP with the same user.

Assuming that both portal servers reside in the same domain, a request to SAP EP transfers the logon cookie for WSP. The new login module uses this cookie to send an HTTP request to a dedicated WSP portlet that returns the user ID currently logged on to WSP. The login module uses this user to log on to SAP EP.

In the case that both portals reside in different domains, it is important to get the relevant cookie values from the URL when initiating the first call to SAP EP. These values are used by the SAP EP login module for logging into WSP as well as for creating new cookies in the domain of the SAP EP side.



4 Details of Solution

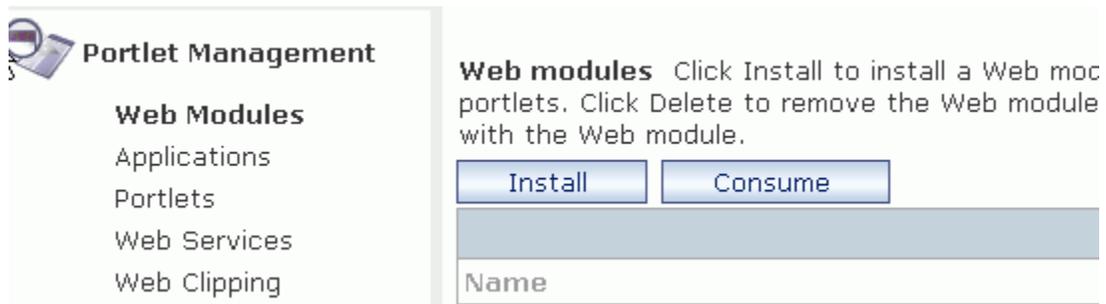
Four steps have to be considered for this solution:

1. Developing and deploying the WSP portlet, which returns the user ID.
2. Developing and deploying the SAP EP JAAS module.
3. Configuring SAP EP JAAS module parameters in the Visual Admin tool of the SAP J2EE Engine.
4. Configuring SSO and the session parameter in the WSP Administration Console.

4.1 Developing and Deploying a WSP Portlet

The following steps describe the deployment of the needed WSP portlet. It will return the username of the currently logged in user within the HTTP header. You can find the pre-compiled WAR file with source code [here](#).

Step 1: Install the WAR file as a web module (Administration/Portal Management).



The screenshot shows the 'Portlet Management' interface. On the left, there is a navigation menu with 'Web Modules' selected. The main area displays 'Web modules' with instructions: 'Click Install to install a Web module portlets. Click Delete to remove the Web module with the Web module.' Below the instructions are two buttons: 'Install' and 'Consume'. A table with a header 'Name' is visible below the buttons.

Step 2: After a successful installation, you will find following entry in the section "Portlets."



The screenshot shows a portlet entry in the SAPSSOManager. The entry is labeled 'Portlet SAPSSOManager' and includes a set of icons for configuration, refresh, edit, delete, and help.

Step 3: Now change the properties of this portlet – at least the SenderIP parameter. Set the value of SenderIP to the IP of SAP EP.

Parameters and Values Enter a new parameter and value pair in the blank fields to create a new parameter for this portlet, or click Delete to remove a parameter and value. Click OK to keep your changes or Cancel to quit.

New parameter: New value:

Page 1 of 1	
Parameter	Value
SenderIP	192.168.0.21
HeaderSAPUID	com.ibm.wps.portlet.sapssomanager.SAPUserid
HeaderRC	com.ibm.wps.portlet.sapssomanager.RC
VaultSlotName	SAPSSOManager
HeaderWPSUID	com.ibm.wps.portlet.sapssomanager.WPSUserid
SecretType	0
HeaderSAPPW	com.ibm.wps.portlet.sapssomanager.SAPPW

Page 1 of 1

Step 4: You can also create a URL mapping to the portlet – it which will give you a shorter URL to enter in the JAAS module (optional).

URL mapping portlet

Root contexts

Root URL contexts add new contexts, click map icon to create or edit mapping, click edit to edit context, or delete to delete context

Page 1 of 1	
Context Label	Mapped to
SSOWebsphere	SAPSSOManager

Step 5: Verify that the SAPSSOManager Portlet is working (if you use URL mapping, for example by launching /wps/myportal/SSOWebsphere). It is very important that all users have access to it. When you call the portlet it should print the name of the user currently logged in.

SAP SSO Manager

This portlet is configured to return a SAP userid for the authenticated WebSphere user.

Current logged in user is: wpsbind

4.2 Setting Up the SSL Keys and Certificates on the WSP (Optional)

You can find detailed information on how to install self-signed keys and certificates in the IBM technote, "[Creating Custom Secure Socket Layer \(SSL\) Key Files for V5.](#)"

4.3 Setting Up the SSL Keys and Certificates on SAP EP (Optional)

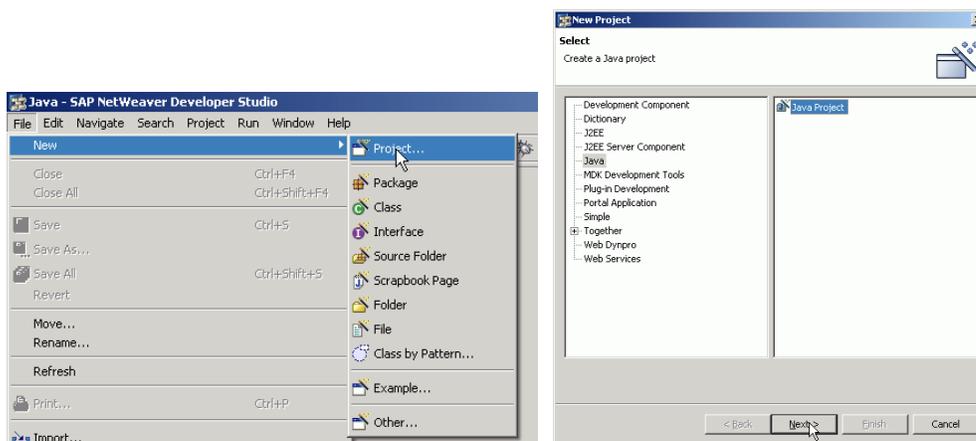
If you use your own CA for the WSP certificate, you must import it into the Java Keystore "cacert." You can export the certificate as Base64 encoded file with the Ikeyman tool. You install it afterwards with the following command in SAP EP:

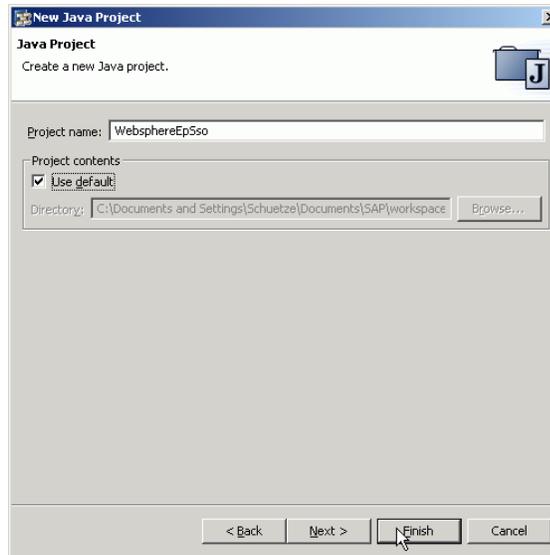
```
keytool -import -file cert.arm -keystore c:\usr\sap\<SID>\jdk...jre\lib\security\cacerts -
alias "WSP_CA" -trustcacerts
```

4.4 Developing and Deploying the SAP EP JAAS Module

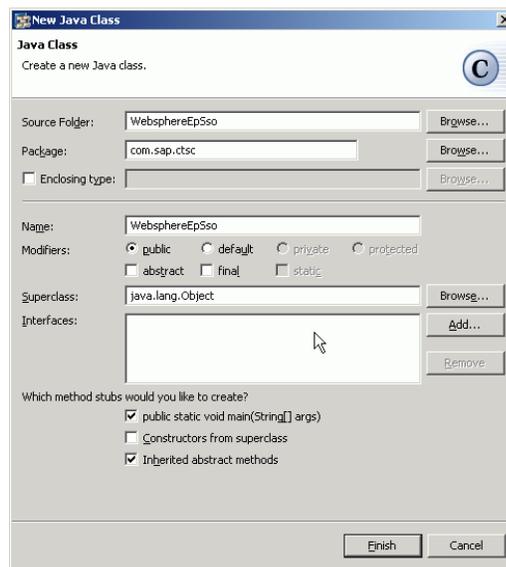
The following steps describe the creation of the required SDA file with a JAAS module. This allows you to change the Java source code if needed. You can also use the pre-compiled SDA file [here](#).

Step 1: Start the SAP NetWeaver Developer Studio and create a new Java Project. Assign a project name, for instance "WebsphereEpSso."





Step 2: Create a new Java Class within this project. Give it a name such as “WebsphereEpSso” and a package named “com.sap.ctsc.”



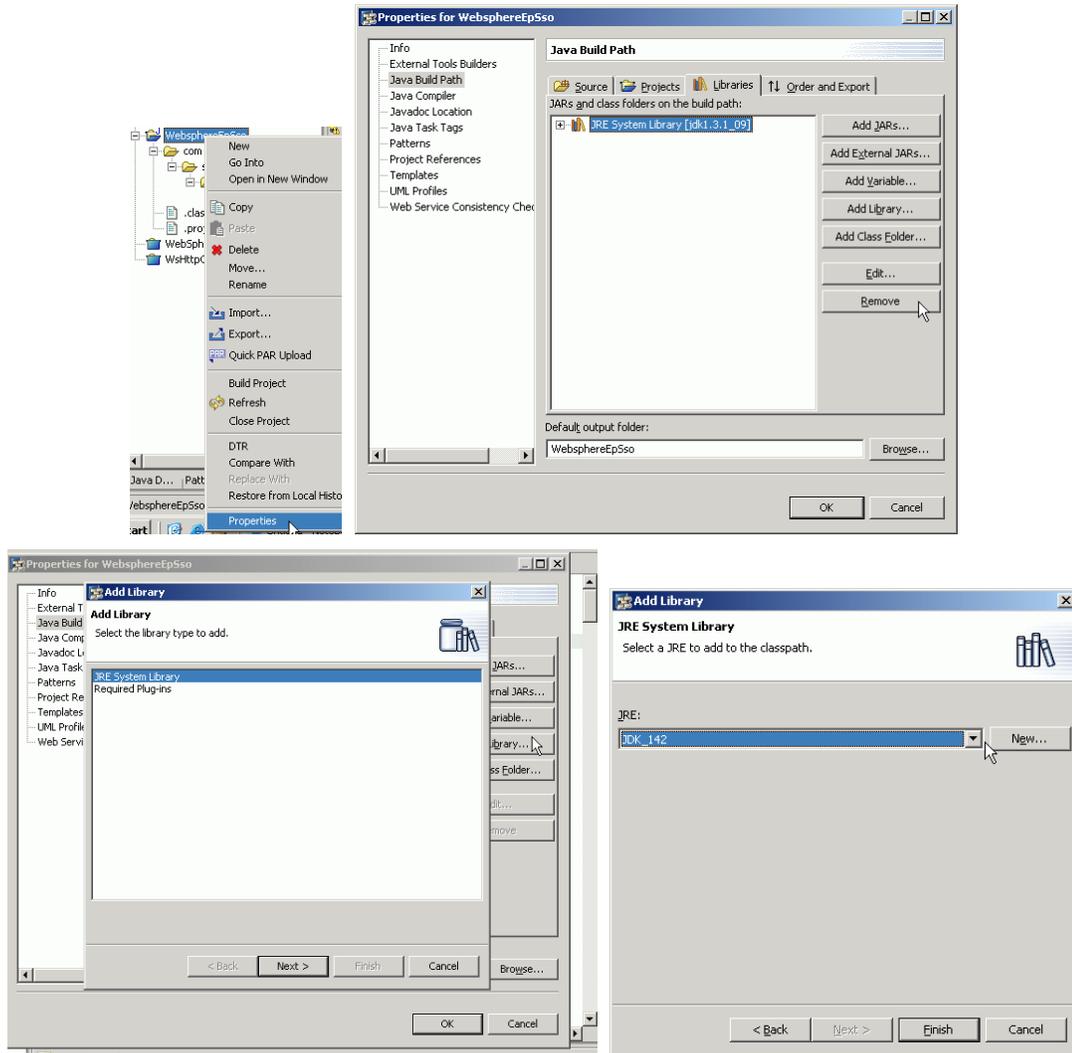
Step 3: Change the properties of the project to import the missing libraries. You will need logging.jar, security_api.jar and util.jar from the SAP EP directory. The logging.jar and util.jar can be found under the following directory:

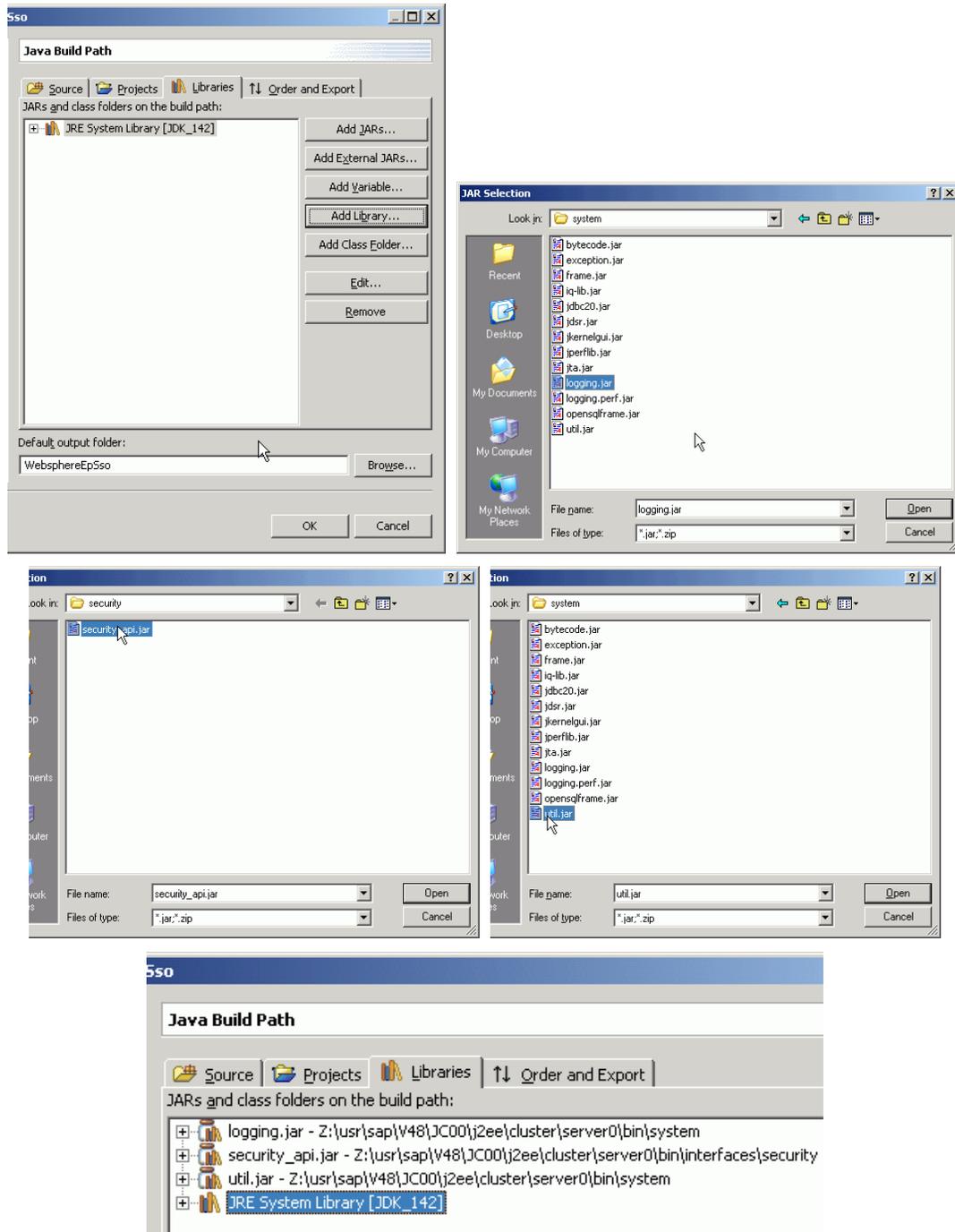
<driveletter>:\usr\sap<SID>\JC<instance#>\j2ee\cluster\server0\bin\system.

The security_api.jar is located under:

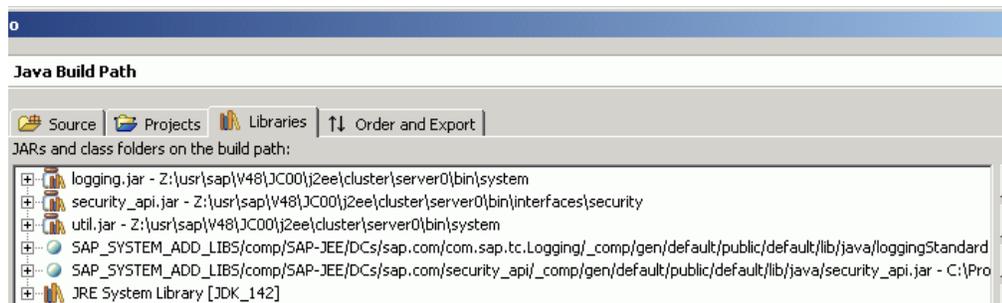
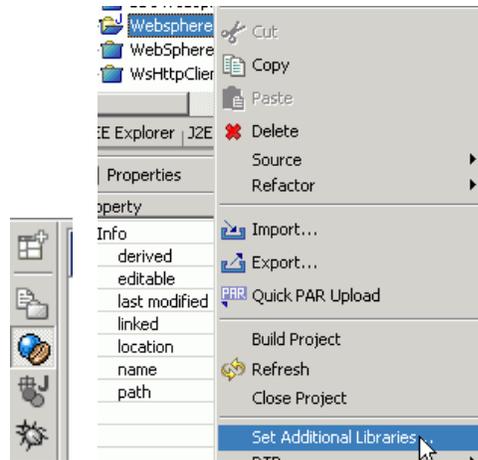
<driveletter>:\usr\sap<SID>\JC<instance#>\j2ee\cluster\server0\bin\interfaces\security.

Also make sure that you use the JDK 1.4.x library (JDK 1.3.x won't work).

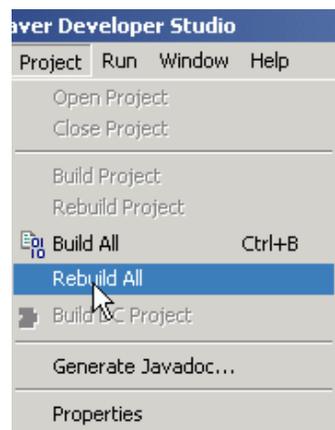




Step 4: Now change to the J2EE Perspective and right-click on the project. Click Set Additional Libraries and mark the entries “com.sap.tc.logging” and “security_api.”



Step 5: Now you can rebuild the project and all dependency errors should disappear.

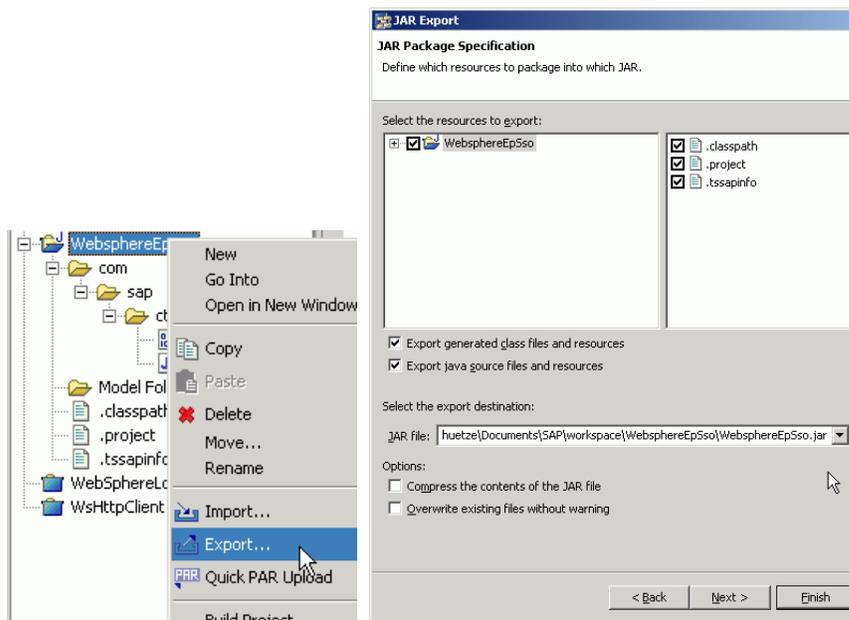


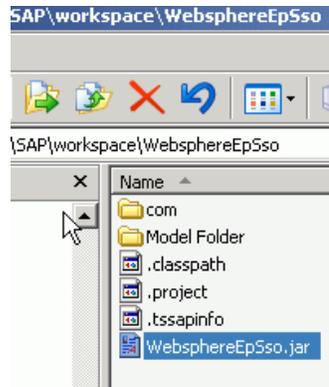
Step 6: Write the necessary coding.

- The class has to extend the AbstractLoginModule class. Therefore the following methods have to be implemented: initialize(), login(), commit(), abort() and logout(). Furthermore four other methods are implemented: getRequestValue(), getCookieValue(), setCookieValue(), getUsernameFromWebSpherePortal().

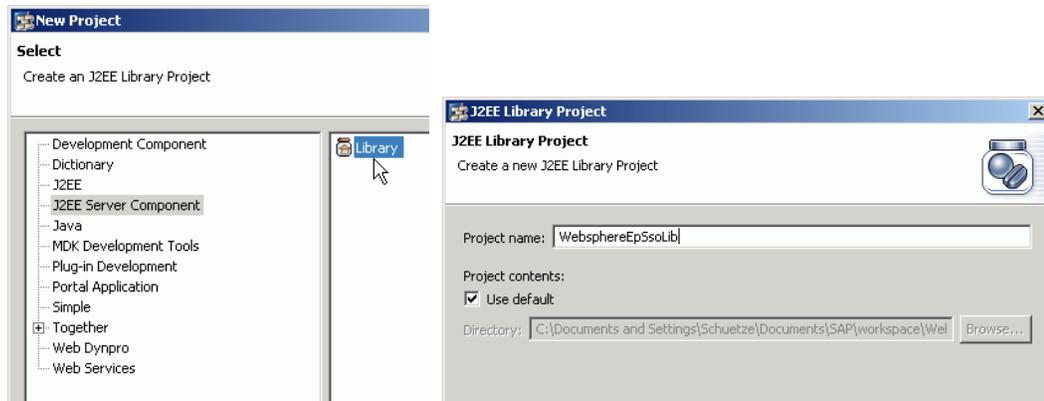
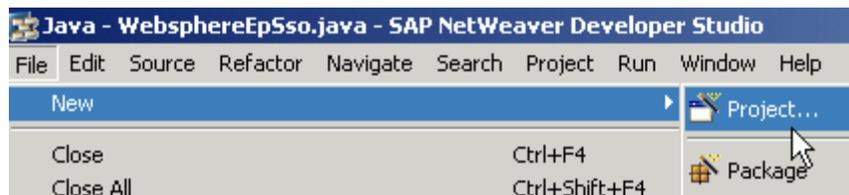
- For the first five methods please refer to the standard SAP documentation on login modules.
- `getCookieValue()`: Returns the value of a given cookie. A callback handler is used, which gets the necessary values from the HTTP request created by accessing the WSP. For further information about `HttpGetterCallback`, refer to the SAP Javadoc.
- `setCookieValue()`: Set the cookie values, which should be available in another domain. A callback setter is used for this purpose. Further information is provided in the `HttpSetterCallback` SAP Javadoc.
- `getRequestValue()`: Returns the value of a requested parameter such as the values of the LTPA and the session cookie.
- `getUserNameFromWebSpherePortal()`: Gets the cookie values, creates a connection to the WSP, and returns the value of the user ID contained in the cookies.

Step 7: Right-click the project and choose Export - “JAR – File.” Now refresh the Project View (F5) – you should see the Jar File within the Project.

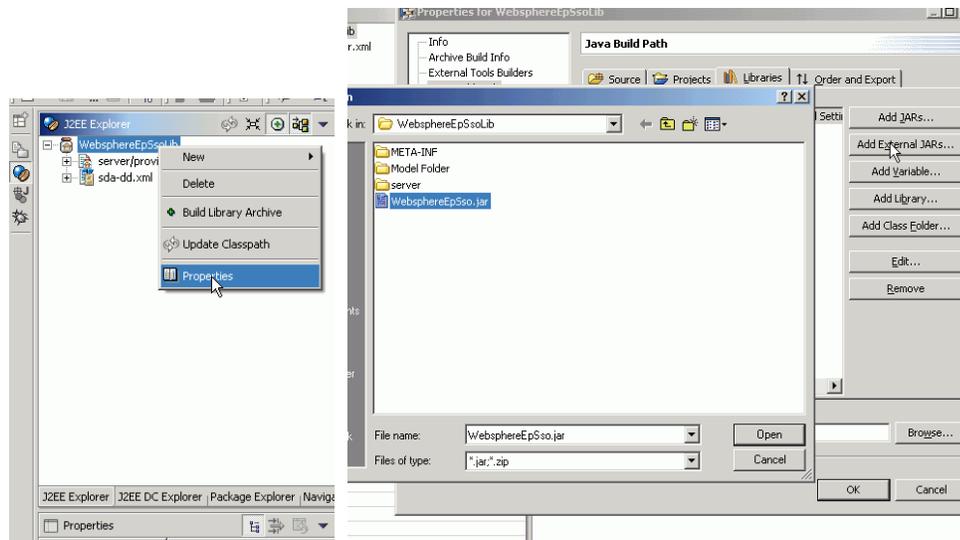




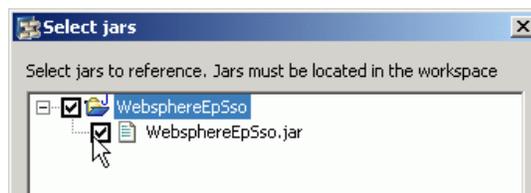
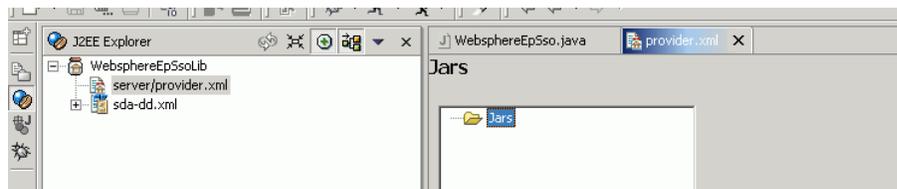
Step 8: Create a new J2EE Server Component Project with an arbitrary name, for instance “WebsphereEpSsoLib.”



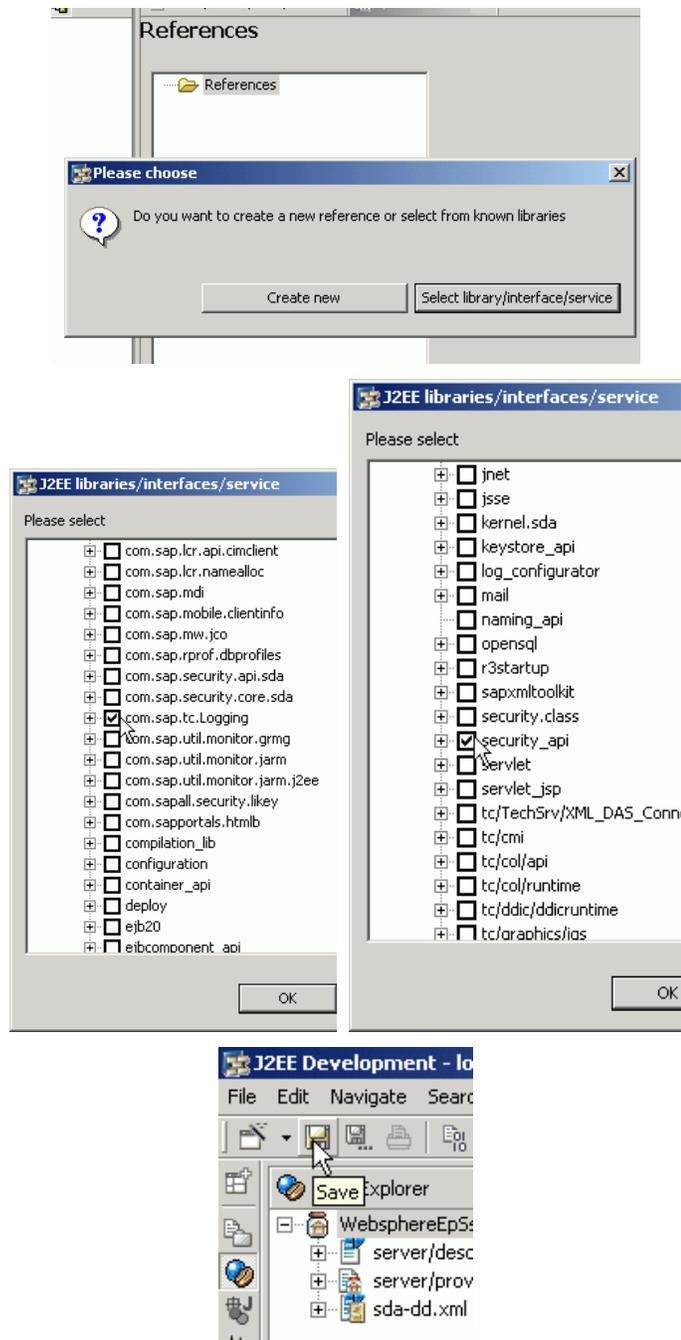
Step 9: Switch to the J2EE Perspective and go the Properties of the WebsphereEpSsoLib project. Go to “Java Build Path” and add the Jar File with the function “Add External JARs.” You can also copy the JAR File to your Lib Project Directory and use the function “Add Jar.”



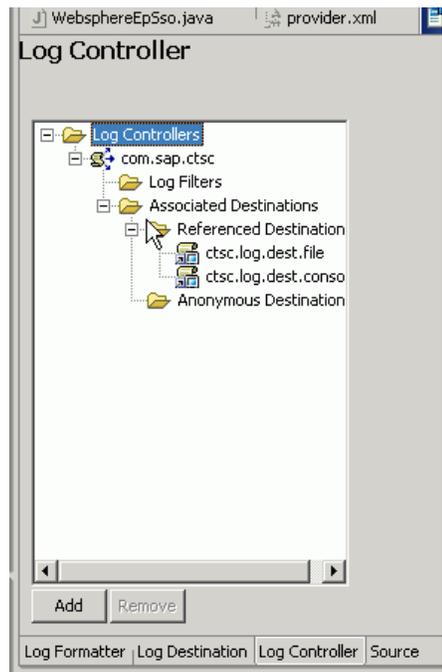
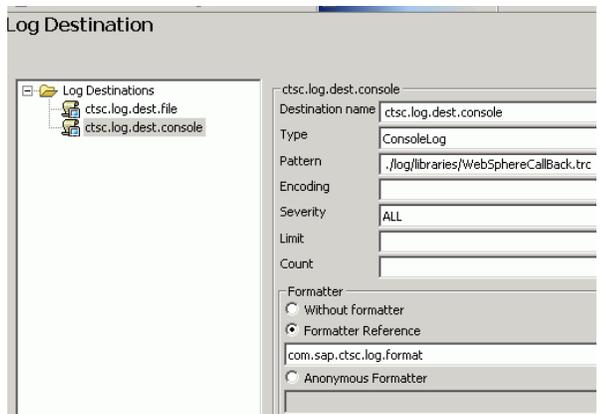
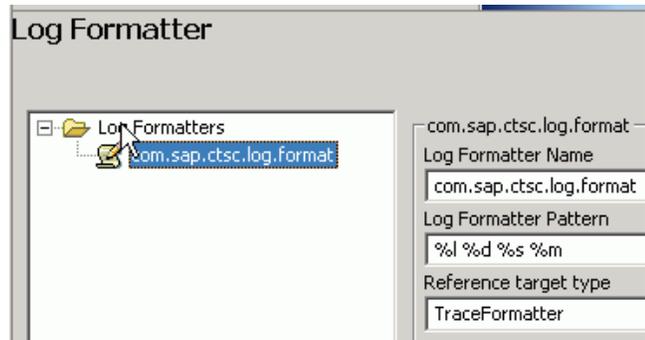
Step 10: Double-click the provider.xml and choose the “Jars” tab. Click the folder icon and select Add. Afterwards mark the WebsphereEpSso.jar and click OK.

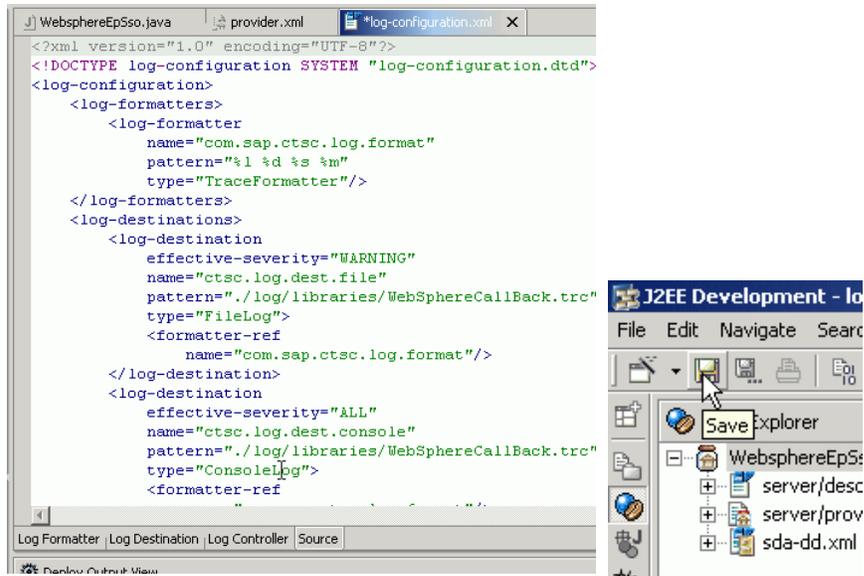


Step 11: Click on the References tab and select “Select library/interface/service.” Mark the entries “com.sap.tc.logging” and “security_api.” Make sure you apply the Save button after you have made your changes.

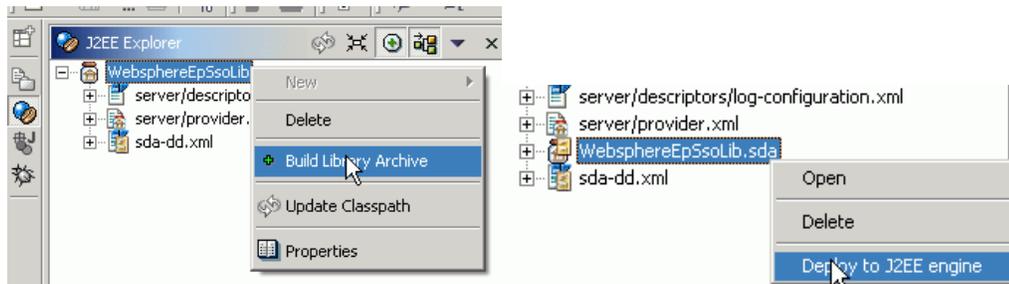


Step 12: Right-click the project and choose New / “server/descriptors/log-configuration.xml.” You can click on the Source tab and paste the content of the supplied log-configuration.xml file or you can define your own destinations/formatters/controllers. Click the Save button after your changes.



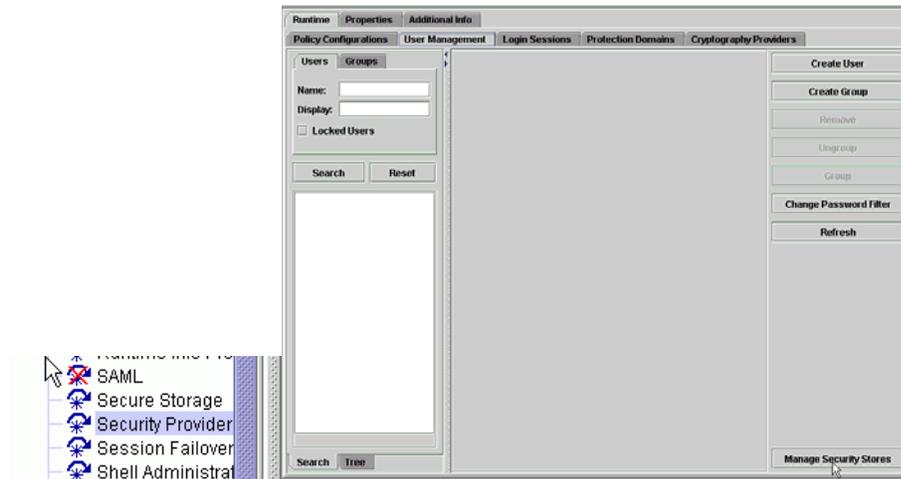


Step 13: Right-click the project and choose “Build Library Archive.” A WebsphereEpSsoLib.sda will be created in the directory of the library project. You can deploy this file with a right-click on it (or you can use the SDM).

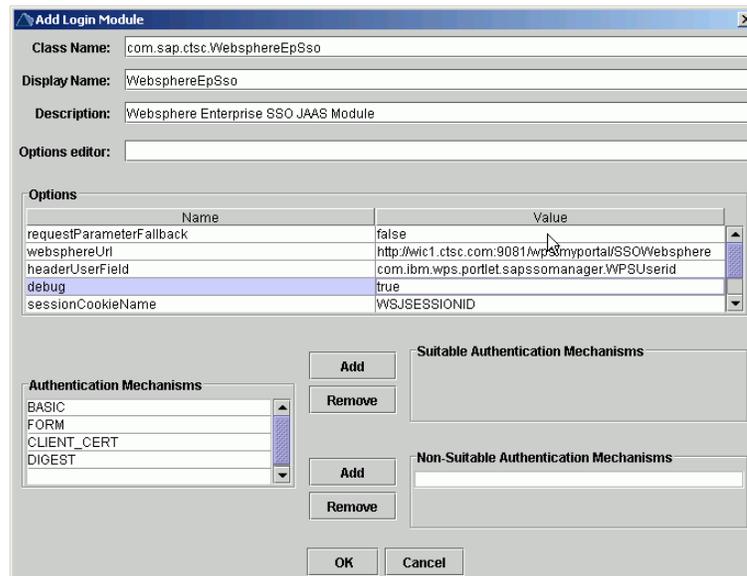
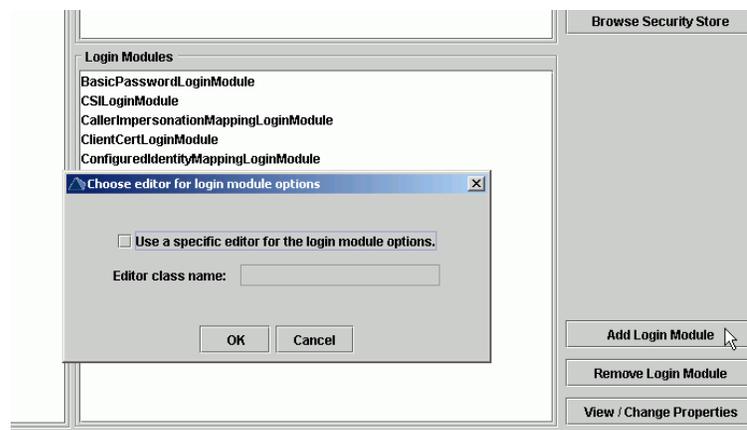


4.5 Configure the SAP EP JAAS Module

Step 1: Start the Visual Administrator of the SAP J2EE Engine and start the Security Provider (located under SID/ServerX/Services). Click the Runtime tab and then choose User Management. Click the Manage Security Stores button.



Step 2: Click the Add Login Module button. Enter the class name (the Java class from the WebsphereEpSso Project), display name, and a description.



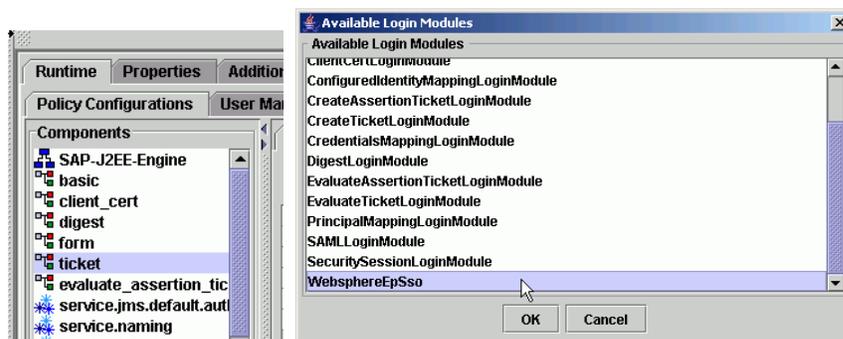
Now enter the options for the JAAS module:

ltpaCookieName	Name of the WSP Ltpa cookie LtpaToken
debug	Debug mode true / false If true you will find debug information in the console log and in the trace file.
sessionCookieName	Name of the WSP session cookie WSP default is JSESSIONID – but you should change the name in Admin Pages of the Websphere Portal (e.g. to WSJSESSIONID).
headerUserField	Name of the header field with the username. Default value of the WSP SSOManager Portlet: com.ibm.wps.portlet.sapssomanager.WPSUserid
websphereUrl	Complete URL to the SSO Portlet on the WSP (a URL mapping helps to get a friendlier URL for the SSOManager Portlet). http://example.ctsc.com:9081/wps/myportal/SSOWebsphere You can also enter a HTTPS site https://example.ctsc.com:9444/wps/myportal/SSOWebsphere
secure	Is a HTTPS connection used true / false This parameter should be true if you use a HTTPS url for “websphereUrl”.
requestParameterFallback	Parameter fallback for working in different domains true / false If true the module will try to get the Ltpa and the session values from the request parameters. Example: ?LtpaToken=123456&WSJSESSIONID=654321 But make sure that you encode the cookie values with the URLEncoder class (@ → %40 ...).
decodeRequestParameter	Decode the incoming request parameter values true / false

	<p>If true the module will try to decode the incoming request values. That can be necessary in some cases when you see missing characters in the log file.</p> <p>Only used if "requestParameterFallback" is set to true</p>
setCookieBothDomains	<p>Set WSP cookies also in the EP domain</p> <p>true / false</p> <p>If true the module will set the WSP token values as cookies also in the EP domain. You should only do this if you have changed the name of the WSP session cookie (e.g. to WSJSESSIONID).</p> <p>Only used if "requestParameterFallback" is set to true</p>

Step 3: Click on the Policy Configurations tab and choose the ticket component. Add the “WebsphereEpSso” module and add another “CreateTicketLoginModule” entry. Also change the flags and the order of the modules (click the Modify button).

The EvaluateTicketLoginModule should be set to “Sufficient,” followed by the WebsphereEPSso login module, which was described in the previous chapter. If the user can be authenticated using this module the following CreateTicketLoginModule creates the logon ticket for the user and the corresponding SAP cookies. Thus the WebsphereEpSso should be set to “Optional” and the CreateTicketLoginModule to “Sufficient.” If it fails the logon request is passed on to the BasicPasswordLoginModule, to which the user can pass his credentials manually and if the authentication is successful CreateTicketLoginModule is finally able to create the logon ticket. Since the BasicPasswordLoginModule needs to succeed, it has to be set to “Requisite,” otherwise the user will not be able to logon to SAP EP and control is immediately returned to the WSP. CreateTicketLoginModule is set to “Optional” in this case.



Authentication	Security Roles	Resources
Authentication template: no		
Login Modules	Flag	Options
com.sap.security.core.server.jaas.EvaluateTicketLoginMod...	SUFFICIENT	{ume.configuration...
WebsphereEpSso	OPTIONAL	{ltpaCookieName=L...
CreateTicketLoginModule	SUFFICIENT	{}
BasicPasswordLoginModule	REQUISITE	{}
com.sap.security.core.server.jaas.CreateTicketLoginModule	OPTIONAL	{ume.configuration...

Step 4: Click on the Properties tab and change the “LoginModuleClassLoaders” Value to “library:WebsphereEpSsoLib.” Click the Save button after your changes. An SAP EP restart is required for the changes to take effect

Runtime	Properties	Additional Info
Key	Value	
CaseSensitive	true	
LoginModuleClassLoaders	library:WebsphereEpSsoLib	
MaxFileLength	1048576	
MaxLogCount	100000	

4.6 Debug the SAP EP JAAS Module

If you encounter any problems, you should change the “debug” parameter of the JAAS module to “true.” The debug messages will be logged to the trace file and the server console (both destinations are configured by the log-configuration.xml). You can use the LogViewer of the Visual Administrator to view the trace file.

Note: you will find the output in the defaultTrace.trc file if the LogManager parameter “ForceSingleTraceFile” has the value “YES.”

4.7 Configuring SSO and Session Parameter in WSP Administration Console

SSO Configuration

In order for SAP EP to receive the login cookie from the browser in the session, SSO must be configured in WSP.

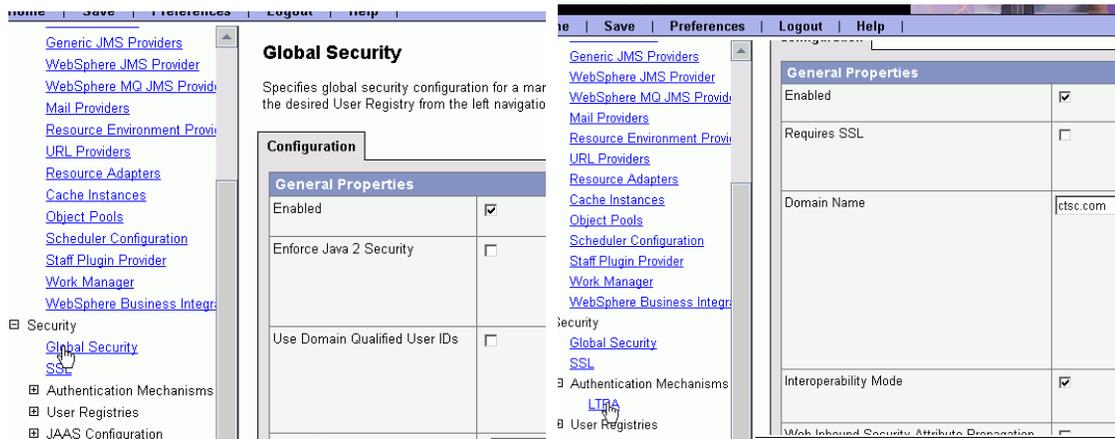
Step 1: Start the WSP Administrative Server for the administrative domain.

Step 2: Start the WSP Administration Console.

Step 3: In the Administration Console, select Security Center from the console menu.

Step 4: Click on the Authentication tab. Adjust the following settings:

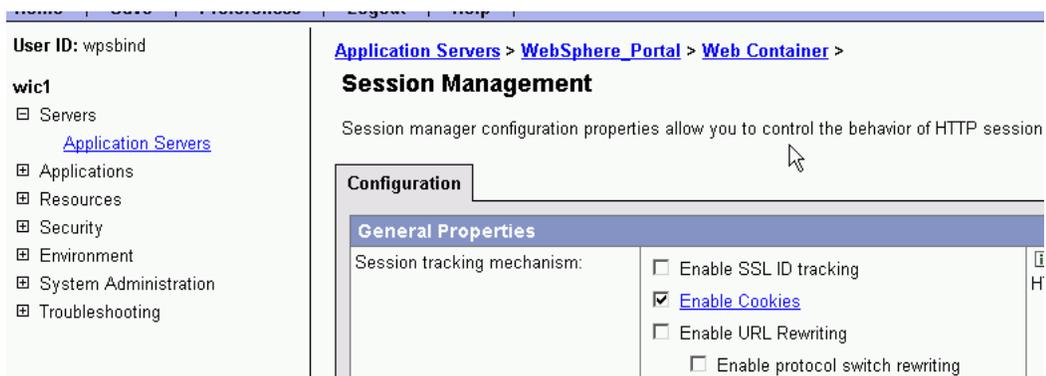
- Set the Authentication field to Lightweight Third Party Authentication (LTPA) in order to use an LDAP directory as the user registry.
- Check the Enable Single Sign-on (SSO) box in order to enable SSO and authentication information to be placed in HTTP cookies.
- Set the Domain field to the domain portion of your fully qualified DNS name for the system running your WebSphere application server administrative domain. For example, if your system's host name is "myhost.mycompany.com," type "mycompany.com" in this field.



For more information on this topic, see [IBM's documentation](#).

Configuring the Session Parameter

Make sure that you change the name of the WebSphere session cookie to another arbitrary name, different from the default "JSESSIONID," which is also used on the SAP EP side, for instance to "WSJSESSIONID." Otherwise you will run into problems since the WebSphere session cookie would overwrite the cookie set by SAP EP.



Cookies

Specify cookie settings for http session management. ⓘ

Configuration

General Properties		
Cookie name	<input type="text" value="WSJSESSIONID"/>	ⓘ A unique name for the cookie to be used for session management. The servlet specification requires this name to be JSESSIONID. However, for flexibility this value is configurable.
Secure cookies	<input type="checkbox"/> Restrict cookies to HTTPS sessions	ⓘ Whether session cookies include the secure field. Enabling the feature will restrict the exchange of cookies only to HTTPS sessions.

Secure cookies	<input type="checkbox"/> Restrict cookies to HTTPS sessions
Cookie domain	<input type="text" value="ctsc.com"/>
Cookie path	<input type="text" value="/"/>