Oracle FLEXCUBE Private Banking
26-September-2011

Generic SSO Adaptor – Configuration and Development

Revision History

Date	Version	Description	Author
26-Sep-2010	1.0	Initial Document	VJ

➢ Introduction

The following document details the development/configuration requirements that would be required to setup a generic SSO adapter for the Oracle FCPB Product

Please note that this is just a development guide and a basic template to achieve the SSO integration with any third party tools or custom development. This may require additional code/configuration to make it work as per the individual site requirements

➢ FLOW DESCRIPTION

Suppose an external system sends us the

http://10.180.126.31/fcpbbank/ssoLogin.action

This URL contains data for the following attributes

The processing first hits the filter "PBSExternalAuthenticationProcessingFilter" which helps in retrieving the data from the URL and are stored as session attributes. This filter will read the login information which will be username for FCPB from the header as below

```
String loggedInUser = httpServletRequestObject.getHeader("Login");
```

This is just a sample parameter that is used in this code. You can customize the code as per your requirements and accordingly retrieve the other parameters as well make a custom call to the master system to validate the authenticity of this request. This filter is hit only once during the login process and hence you can use this as the entry point validation for the system

The most important thing to note out is that to enable SSO in Oracle FCPB, it requires an entry for that user in the SMS_APP_USER table. So in case you have a customer upload interface, you would need to ensure that this entry gets created when the customer is created. The default interface provided by the product for the customer upload has this facility provided. The user <-> role mapping is done as per the segment i.e. the segment name is the same as that of the role.

IF THIS IS NOT SETUP DURING THE INTERFACE UPLOAD, THE WHOLE SSO OPERATION WILL FAIL

The following product jar files are required in case you would want to extend the same

- wminfra-2.0.1.jar
- wmcrmcore-2.0.1.jar and
- wmsmscore-2.0.1.jar

Please note that in addition to the above, you would need the following external libraries to make this work

- Spring 2.5.5
- Struts 2
- Hibernate 3.3
- Apache commons
- Apache commons HTTP client

The product team can guide you in setting up the workspace to develop on this adaptor. You would require strong Spring framework knowledge including the Spring security framework to get this working.

➢ CODE BASE/CONFIGURATION

To setup the code base for the SSO adapter, the following things would need to be done

- Setup Eclipse Helios on your local workstation
- Copy the attached jar file to a local folder and ensure that this jar file is setup within your ANT workspace. This jar file contains the tasks for ivy and needs to be done before the build is run. To setup, go to Window -> Preferences -> Ant -> Runtime -> Classpath -> Ant Home Entries -> Add External Jars
  This should ensure that the ant ivy tasks will be executed for the build

  

  ivy-2.0.0-rc2.jar
- Check-out the FCPBOAMADAPTER from Subversion. If you do not have the rights request that the product team provide you the checked-out version of the path. Changes if any would have to be maintained locally and checked-in into the implementation team's version control
- Get the latest repository folder "REPOSITORY" from the product team. This should contain all the jar files that would be required for your development and build purpose
- The REPOSITORY has the following folders
  - CORE
  - FCPBREPOSITORY -> Contains jar files for the ivy build
  - PBIVYREPOSITORY (IVY Repository)
  - LIBREPOSITORY -> Contains jar files for the eclipse build
- Please modify the following files for your build
  - build.properties -> The following properties need to be changed "settings.localRepository" to the path to the folder FCPBREPOSITORYand "ivy.user.dir" to the PBIVYREPOSITORY folder. This is for the ivy build
  - .classpath file -> Replace the path entries for the location of LIBREPOSITORY and PBIVYREPOSITORY. This is for the eclipse build
- The ivy build helps you in creating the jar file for the adapter. Open the build-individual.xml, right-click and run as an Ant build. This will generate the jar file in the ivy repository. This adapter jar can then be put into the WEB=INF/lib folder of your war file to get the adapter working

➢ RUNTIME CONFIGURATION

Once the jar file is generated, the following runtime configuration needs to be done so that the adapter is working ok

The following changes need to be done in the web.xml file after the main filter for Struts

1. **Adding SSO Spring security file (WmSSOSpringWebSecurity.xml) to spring context**

   Replace

   /WEB-INF/WmSpringWebSecurity.xml

   With

   classpath*:/WmSSOSpringWebSecurity.xml

2. **Adding Session validation filter**

```
    <filter>
      <filter-name>externalLoginActionFilter</filter-name>
      <filter-
class>com.iflexsolutions.fcpb.efinacle.filter.PBSExternalLoginActionFilter</
filter-class>
    </filter>
    <filter-mapping>
        <filter-name>externalLoginActionFilter</filter-name>
        <url-pattern>*.action</url-pattern>
    </filter-mapping>
```

3. **In the file wmStruts-sms.xml which will be in the file wmsmsweb-2.2.jar, you need to add the following entry**

```
<action name="oamLogin" class="oamLoginAction">
      <interceptor-ref name="exception"/>
      <interceptor-ref name="alias"/>
      <interceptor-ref name="servlet-config"/>
      <interceptor-ref name="prepare"/>
      <interceptor-ref name="i18n"/>
      <interceptor-ref name="chain"/>
      <interceptor-ref name="wmRole"/>
      <interceptor-ref name="wmRoleFunction"/>
      <interceptor-ref name="debugging"/>
      <interceptor-ref name="profiling"/>
      <interceptor-ref name="scoped-model-driven"/>
      <interceptor-ref name="model-driven"/>
      <interceptor-ref name="checkbox"/>
      <interceptor-ref name="static-params"/>
      <interceptor-ref name="params">
            <param name="excludeParams">dojo\..*</param>
      </interceptor-ref>
      <interceptor-ref name="conversionError"/>
      <interceptor-ref name="validation">
```

```
                  <param
name="excludeMethods">input,back,cancel,browse</param>
            </interceptor-ref>
            <interceptor-ref name="workflow">
                  <param
name="excludeMethods">input,back,cancel,browse</param>
            </interceptor-ref>
            <interceptor-ref name="pbs-token-session"/>
            <interceptor-ref name="xss-injection"/>

            <result name="next" type="redirect-
action">${nextAction}</result>
      </action>
```

4. **Replace the logout.jsp file found in the war file with the one which is found in the SSO Adapters projects home directory.**

5. **Make below highlighted entry in the decorators.xml file located in the WEB-INF folder of the war file.**

```
<excludes>
…
<pattern>/logout*</pattern>
…

</excludes>
```

6. **You can use the apache commons logging service for your logging. Please make the changes in log4j.xml to include your new packages. This is available in the WEB-INF/classes for your war file. Please do not as a practice put in System.out.println(“”) in your code base. The log factory class can be obtained like this**

```
// Logging Imports
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;

// Within the class file
protected final Log log = LogFactory.getLog(getClass());
```