Application Decommission SiteMinder Integration

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Revision | Contributors | Summary of Changes |
| 24-April-2016 | 1.0 | Malik | Initial Version |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1. Summary 4](#_Toc449221474)

[2. High Level Architecture 5](#_Toc449221475)

[2.1.1. Default User Authentication Model 5](#_Toc449221476)

[2.1.2. Default Authorization Model 5](#_Toc449221477)

[2.1.3. SiteMinder Security Model 5](#_Toc449221478)

[3. Migrating Applications to use SiteMinder Authentication and Authorization 7](#_Toc449221479)

[4. Help 8](#_Toc449221480)

# Summary

This document discusses changes made to the XML Application Decommissioning Framework to support SiteMinder integration.

# High Level Architecture

### Default User Authentication Model

The decomm framework is configured with a DDS service that handles user authentication requests. The default user service delivered with the framework provides xDB local authentication and authorization. A user name/password set of credentials are entered into a login dialog and the decomm client code performs an HTTP POST operation to pass this information to the user service. The user service in turn encrypts the user password credential and then compares it against an encrypted password stored in xDB for the specified user. Subsequent to a successful user authentication, an HTTP header is set indicating that the user now has a login session with the server. All subsequent requests to any DDS/xDB service utilize the DDS token service to confirm that the user’s login token is still valid. On session timeout, the token service returns information indicating that the session has timed out. In this situation, the decomm client code redirects to the user login dialog for user authentication.

### Default Authorization Model

The decomm framework also provides an authorization model that is based on the user’s role for a particular application. In the default implementation, decomm client code uses the DDS XQuery service to locate an XML file within the xDB database that represents the user. This file has a role id element that is used to access information stored in xDB in a role file that describes the user’s role within the application. The role information returned from this xQuery is cached in the decomm client layer. This role information is subsequently used to restrict behavior in areas such as data sets that are visible to the user, operations and query results.

### SiteMinder Security Model

The decomm SiteMinder security model is implemented through a new DDS user service, role service and some changes in the decomm client code that provide an SSO style authentication model if the user has previously been authenticated using a SiteMinder recognized authentication mechanism.

The decomm SiteMinder functionality is enabled by modifying the web.xml file in the application’s WEB-INF/classes directory and also the services.xml file stored in the application’s xDB Application/resources library to indicate to the decomm client classes that the new user service should be called. The authentication and authorization algorithm is as follows:

* In main.java, before the login dialog is displayed, onUserServiceConfigurationCheck() is called to confirm that the user server is available. UserServiceAsync.login() is called with empty user name/pw credentials to see if the browser session has already been authenticated by SiteMinder
* The HTTP Post request to the new UserService implementation (XMLArchivingUserGWTServiceDelegate) is intercepted by the SiteMinder security module running on the Apache httpd server.
* The SiteMinder security module inspects the URL requested in the POST message. If it is a SiteMinder protected resource, the SiteMinder module determines if the browser session has authentication credentials. In the UHG MS domain environment, this is done by using Windows NTLM authentication.
* If the user is not authenticated, the request is directed to a generic page indicating that the user must login to the domain before accessing the requested resource
* If the user is authenticated, the SiteMinder security module checks to see what AD groups the user is a member of. It checks this list against a list of groups that are authorized to access the resource.
* The request is forwarded to the XMLArchivingUserGWTServiceDelegate servlet, if the user is both authenticated and authorized,
* The XMLArchivingUserGWTServiceDelegate.login() method is called and passed an empty string for both the login user and password strings.
* The XMLArchivingUserGWTServiceDelegate.login() method looks at the HTTP headers passed in the request. From the headers the required values are obtained.
* The Java resource property file smauth.properties available in the application’s src folder is accessed and the property com.uhg.xmlarchiving.smauth.groupNames is accessed to determine the base name of groups to be used when looking up the group name that represents a role for the current application.
* The HTTP header is parsed and the group name that contains the string specified in the groupNamePrefix property. This represents the role that the user has for this application. The role information is stored in the XMLArchivingUser object for subsequent use by the role service.
* If the SiteMinder login is successful:
* The client code suppresses the login dialog display and uses the returned user information to call the role service.
* The role service returns the role associated with the user
* The client code retrieves the role id from the role object returned from the role service and uses the DDS GWT query service to search for an XML file in the xDB database that represent the serialized version of the role.
* A DDS role object is created from information contained in the role XML file. This role object can contain restrictions that limit the user’s access to data nodes, operations and search results.
* The role object is cached in the GWT client context so that subsequent queries can utilize the role to restrict search results based on role.
* If the SiteMinder login was unsuccessful:
* The login dialog is displayed to collect login credentials
* The user service is used to login the user. Since SiteMinder integration is not present, the user service does local xDB authentication using the user credentials provided in the login dialog. When this type of authentication is performed, the XMLArchivingUser is created with a flag indicating that the role information should be retrieved directly from xDB by the client. (This flag will be removed in a future iteration of this code and role loading will occur on the server side for xDB authentication as well as SiteMinder authentication)
* After successful login, the client calls the role service to determine the role.

# Migrating Applications to use SiteMinder Authentication and Authorization

The following steps should be performed to migrate an application to use the SiteMinder Authentication and Authorization functionality.

* Generate the smauth.jar file
* Place this jar into the lib folder within your application
* Modify the Eclipse project setting and the build.xml file to reference smauth.jar as a dependent library. Refer to the Oasis project for an example of these project settings.
* Modify the web.xml file in your war/ WEB-INF/ directory as follows:
  + Add a definition for the new role servlet:

<servlet>

<servlet-name>RoleService</servlet-name>

<servlet-class>com.emc.dds.xmlarchiving.server.auth.RoleServiceImpl</servlet-class>

</servlet>

* + Add the mapping for the new role servlet:

<servlet-mapping>

<servlet-name>RoleService</servlet-name>

<url-pattern>/oasis/RoleService</url-pattern>

</servlet-mapping>

* + Change the user service definition to reference the new user service

<servlet>

<servlet-name>UserService</servlet-name>  
 <servlet-class>com.emc.dds.xmlarchiving.server.auth.XMLArchivingUserGWTServiceDelegate</servlet-class>  
 </servlet>

* In xDB, modify the file /APPLICATIONS/<app\_name>/configuration/Services.xml to point to the new user service - com.emc.dds.xmlarchiving.server.auth.XMLArchivingUserServiceDelegate.
* In xDB, change role files in the directory tree /APPLICATIONS/<app\_name>/roles to match the role definitions in AD. For each role in AD, you should have a file in the /APPLICATIONS/<app\_name>/roles directory that has an id element that matches the group name.
* Configure SiteMinder access policies so that the appropriate URI for the application is protected by SiteMinder and that only users in the appropriate groups are given permission to access the resource.
* Deploy and test the SiteMinder integration. Information on SiteMinder access for authentication is logged to catalina.out to assist in debugging deployment issues.

# Help

The below attachment describes how to generate the smauth jar file and how to retrieve user id from site minder to EMC application.

