

# Fuego Enterprise 5.1 and 5.5

## Work Portal Authentication

using

## Microsoft Active Directory

Date: April 17th, 2005  
Version: Final  
Author: Eduardo Chioconni  
Title: Director of Product Management

## Table of Contents

Table of Contents .....	2
Fuego Enterprise 5.1 and 5.5 Authentication with Microsoft Active Directory.....	3
Introduction.....	3
Configuring Fuego Enterprise 5.1 and 5.5 Embedded Tomcat's Web Server.....	3
Defining Tomcat's JNDI Realm .....	3
Considerations about server.xml.....	3
Considerations about the JNDIRealm tags configuration.....	4
Linking Tomcat's Authentication with Fuego's Container Authentication .....	5
Associating JNDI Realm with Fuego Enterprise's Portal Web Application .....	5
Considerations about the security tags in Work Portal's web.xml .....	6
Configuring Fuego Enterprise Work Portal to use Container Authentication .....	6
Configuring Directory Properties File .....	7
Configuring Fuego Enterprise 5.1 and 5.5 Directory Service to trust Microsoft	
Active Directory authenticated Users .....	8
Configuring Fuego Directory Service Repository .....	8
Testing the Fuego Work Portal authentication delegated to Tomcat's JNDIRealm...	9
Microsoft Active Directory User Replication in Fuego's Directory Service .....	10
Configuring Work Portal's web.xml.....	10
Configuring Work Portal's userauthentication.properties configuration file .....	10
Troubleshooting .....	14
Appendix A.....	17

# Fuego Enterprise 5.1 and 5.5 Authentication with Microsoft Active Directory

## *Introduction*

The following document describes how Fuego Enterprise 5.1 and 5.5 Work Portal can leverage authentication capabilities provided by Microsoft Active Directory Service.

It will explain how to configure Fuego Enterprise Embedded Tomcat Web Server to connect to Microsoft Active Directory and at the same time automatically propagate this user into the Fuego Portal as a trusted one.

In addition to providing the steps to setup this configuration, it also provides troubleshooting tips as well as recommendations for the setup.

## *Configuring Fuego Enterprise 5.1 and 5.5 Embedded Tomcat's Web Server*

### Defining Tomcat's JNDI Realm

In Fuego Enterprise 5.1 and 5.5 Tomcat's server.xml (\$FUEGO\_ENT/tomcat/conf/server.xml) you will need to enter the following Realm tag.

```
<Engine name="Standalone" defaultHost="localhost" debug="0">
...
  <Realm className="org.apache.catalina.realm.JNDIRealm" debug="99"
    connectionURL="ldap://192.168.1.140/DC=scc,DC=labs,DC=fuegotech,DC=com"
    connectionName="Administrator@scc.labs.fuegotech.com"
    connectionPassword="password"
      referrals="follow"
      userBase=""
      userSearch="(sAMAccountName={0})"
      userRoleName="memberOf"
      userSubtree="true"
      roleBase=""
      roleName="cn"
      roleSearch="(memberOf={0})"
      roleSubtree="true"
  />
...
</Engine>
```

### Considerations about server.xml

- Make sure there is not other realm in the server.xml. By default, the “UserDatabase” Realm is enabled. You will need to comment this one out when adding the one specified above.

## **Considerations about the JNDIRealm tags configuration**

- Do not specify any value into the attributes “userBase” and “roleBase” since you will get an Object Not Found error reported by Microsoft Active Directory.
- Use sAMAccountName to identify users even though there may be other attributes that can have the User Id.
- Enable the recursive properties for user and role searches: “userSubtree” and “roleSubtree”.
- Do not omit the “referrals” property set to “follow”
- In this example, we are using the properties “cconnectionName” and “connectionPassword” since the anonymous binding does not permit access to all the participant and role attributes. For this reason, we are using a trusted Microsoft Active Directory user and password for a complete and successful search.
- The user specified in “connectionName” needs to be qualified by the Domain (ie: scc.labs.fuegotech.com).
- If needed, the “debug” tag can be set to “0” after the whole framework works.

## ***Linking Tomcat's Authentication with Fuego's Container Authentication***

This section of the document describes how to plug Fuego Enterprise Work Portal's Container authentication with Tomcat's JNDI Authentication connecting to Microsoft Active Directory.

### **Associating JNDI Realm with Fuego Enterprise's Portal Web Application**

You will need to edit Fuego Enterprise Work Portal's web.xml (\$FUEGO\_ENT/webapps/portal/WEB-INF/web.xml) configuration file and add the following tag at the end of the Web Application definition tag (</web-app>).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
  <display-name>Portal Application</display-name>
  ...
  <security-constraint>
    <display-name>Fuego Default Security Constraint</display-name>
    <web-resource-collection>
      <web-resource-name>
        Protected Fuego Portal by MSAD
      </web-resource-name>
      <url-pattern>/jsp/*</url-pattern>
      <url-pattern>/servlet/*</url-pattern>
      <http-method>GET</http-method>
      <http-method>POST</http-method>
    </web-resource-collection>
    <auth-constraint>
      <role-
name>CN=Administrators,CN=Builtin,DC=sec,DC=labs,DC=fuegotech,DC=com</role-
name>
    </auth-constraint>
  </security-constraint>

  <login-config>
    <auth-method>FORM</auth-method>
    <realm-name>Fuego Work Portal authenticated against MSAD</realm-name>
    <form-login-config>
      <form-login-page>/jsp/loginContainer.jsp</form-login-page>
      <form-error-page>/jsp/loginContainerError.jsp</form-error-page>
    </form-login-config>
  </login-config>

  <security-role>
    <description>Fuego Users</description>
```

```
<role-  
name>CN=Administrators,CN=Builtin,DC=scc,DC=labs,DC=fuegotech,DC=com</role-  
name>  
</security-role>  
</web-app>
```

## Considerations about the security tags in Work Portal's web.xml

- The role name specified in the “role-name” tag needs to be fully qualified. You should not provide just the name of the role.
- We have decided to use a FORM authentication that prompts a customized connection dialog to authenticate the user against Microsoft Active Directory and automatically propagate the User authentication into Fuego's Directory Service.
- In order case, we are mapping the Work Portal Web Application to the Administrators role, however, it should be best practice to define a FuegoRole in Microsoft Active Directory and assign it to all the users that will have access to Fuego's Work Portal. This will be just the authentication and then the security will be enforced by the roles assigned to that person in the Fuego.

## Configuring Fuego Enterprise Work Portal to use Container Authentication

In order to make the user authenticated by Microsoft Active Directory a trusted user to Fuego, we will need to configure Fuego Portal's web.xml file. The following entries:

```
<web-app>  
...  
<servlet>  
  <servlet-name>  
    startup  
  </servlet-name>  
  <servlet-class>fuego.portal.servlet.deploy.SimpleStartup</servlet-class>  
  <load-on-startup>1</load-on-startup>  
</servlet>  
<servlet>  
  <servlet-name>  
    loginWam  
  </servlet-name>  
  <servlet-class>fuego.portal.servlet.deploy.SimpleLogin</servlet-class>  
</servlet>  
...  
<servlet>  
  <servlet-name>  
    logoutWam  
  </servlet-name>  
  <servlet-class>fuego.portal.servlet.deploy.SimpleLogout</servlet-class>  
</servlet>
```

```
...
</web-app>
```

should be replaced with:

```
<web-app>
...
<servlet>
  <servlet-name>
    startup
  </servlet-name>
  <servlet-class>fuego.portal.servlet.deploy.UserPrincipalStartup</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet>
  <servlet-name>
    loginWam
  </servlet-name>
  <servlet-class>fuego.portal.servlet.deploy.UserPrincipalLogin</servlet-class>
</servlet>
...
<servlet>
  <servlet-name>
    logoutWam
  </servlet-name>
  <servlet-class>fuego.portal.servlet.deploy.UserPrincipalLogout</servlet-class>
</servlet>
...
</web-app>
```

## Configuring Directory Properties File

After the Fuego Enterprise Work Portal web.xml configuration file has been completed, you will need to edit its “directory.properties” configuration file. We will need to add 3 new properties related to authentication propagation. These properties are listed below:

```
# Container Authentication Fuego Directory Service Configuration
directory.SQLFDI51.preset.container-auth.jdbc-user=SQLFDI51
directory.SQLFDI51.preset.container-auth.jdbc-password=password
directory.SQLFDI51.preset.container-auth.skip-auth=true
```

- ***directory.DIRECTORY\_ID.preset.container-auth.jdbc-user***: This is the Fuego Directory Service JDBC User to perform container authentication. This user is important for next’s section.
- ***directory.DIRECTORY\_ID.preset.container-auth.jdbc-password***: This is the Fuego Directory Service JDBC Password for the Database user specified in property: `directory.DIRECTORY_ID.preset.container-auth.jdbc-user`.

- *directory.DIRECTORY\_ID.preset.container-auth.skip-auth*: This property should be set to true to automatically login the End User after authenticated by Tomcat's JNDIRealm.

## Configuring Fuego Enterprise 5.1 and 5.5 Directory Service to trust Microsoft Active Directory authenticated Users

This section of the document explains how the Fuego Directory Service needs to be configured to support an authenticated user by a Container just as Tomcat.

The idea behind a trusted user is that ONLY the authentication of the User is delegated to the Container. However, the authorization will still be under Fuego's control.

This implies an implicit synchronization of users in Fuego's Directory Service and the Datasource (in our case Microsoft Active Directory) accessed by the JNDI Realm of the Container .

In the case of deploying the Fuego Directory Service into an RDBMS, we will need to replicate the User Id of the Microsoft Active Directory into a table of the Fuego Directory Service.

This synchronization and configuration is the ones explained below.

## Configuring Fuego Directory Service Repository

In order to provide an automatic login of the End User into Fuego Enterprise Work Portal after authenticated by Tomcat's JNDIRealm, we will need to populate Fuego Directory Services Trusted User Structure. In the case the Fuego Directory Service is deployed into an RDBMS, this structure will be a table. This table name is: "FUEGO\_PARTTRUST". The following tuples need to be inserted into the mentioned table to provide a correct authentication.

Record #1: [fuego_id, fuego_trustid] = [null, SQLFDI51*] Record #2: [fuego_id, fuego_trustid] = [root, SQLFDI51]
---

Let's explain the reason for these 2 records and how they should be interpreted.

**Record #1:** This means that a Fuego Directory Service JDBC User Connection done with the User or Login "SQLFDI51", should trust ANY Fuego Participant already Authenticated against Microsoft Active Directory without doing a double authentication in Fuego. The "\*" postfix means no double authentication on Fuego's Directory Service. The value specified for the fuego\_trustid is the one specified in the property "*directory.DIRECTORY\_ID.preset.container-auth.jdbc-user*" on the Fuego Work Portal directory.properties configuration file.

**Record #2:** This means that a Fuego Directory Service JDBC User Connection done with the User or Login "SQLFDI51" should trust Fuego Participant "root" doing



authentication. We have to have in mind that this “root” participant is consider Fuego’s Administrator participant. The value specified for the fuego\_trustid is the one specified in the property “*directory.DIRECTORY\_ID.preset.container-auth.jdbc-user*” on the Fuego Work Portal directory.properties configuration file. You should note that it does not have the “\*” postfix as we do want to perform authentication for this Fuego participant and the actions executed by it.

For this setup and configuration, these 2 rows are enough.

## **Testing the Fuego Work Portal authentication delegated to Tomcat’s JNDIRealm**

In order to test the authentication, provide the Fuego Enterprise Work Portal URL into your browser. When prompted for a user and password, enter a valid user and password defined for a user in Microsoft Active Directory.

To successfully connect to the Work Portal, the user should also be declared in the Fuego Directory Service even though the password is not checked. Both Microsoft Active Directory User Id and Fuego’s Directory Service User Id must match. This user for this exercise should be created manually.

In the next section, we will describe how we can use a custom Login Servlet that automatically creates and replicates the Microsoft Active Directory User ID if it is not already defined in Fuego.

## ***Microsoft Active Directory User Replication in Fuego's Directory Service***

As described in the previous section, there is a problem to support a totally transparent authentication if the Microsoft Active Directory User does not exist in Fuego's Directory Service. For the user to be authenticated all the way through the Work Portal, it does need to be defined and active in Fuego's Directory Service.

In practical terms, needing to manually replicate each one of the Microsoft Active Directory may be a tedious task. For this reason, this section will describe how Fuego can be configured so that a successfully authenticated Microsoft Active Directory user can be automatically replicated in Fuego's Directory Service.

### **Configuring Work Portal's web.xml**

The login servlet specified in Work Portal's web.xml configuration file needs to be replaced with "fuego.portal.servlet.deploy.UserPrincipalLoginWithUserReplication".

This servlet is available since Fuego 5.1 GA SP3 and on.

```
<servlet>
  <servlet-name>
    loginWam
  </servlet-name>
  <servlet-class>
    fuego.portal.servlet.deploy.UserPrincipalLoginWithUserReplication
  </servlet-class>
</servlet>
```

This servlet executes only after a user has been successfully authenticated against Microsoft Active Directory. The servlet receives the user and password (principal and credential) used to authenticate against Microsoft Active Directory. It checks if the received user id exists in Fuego's Directory Service. If it does not exist, it will try to replicate the User in Fuego's Directory Service. If the user already exists, then it will bypass the user creation and replication in Fuego's Directory Service.

Next section will describe how we can configure the behavior of the user replication servlet.

### **Configuring Work Portal's userauthentication.properties configuration file**

The Login Servlet configured above uses some properties that need to be defined in the Work Portal's userauthentication.properties configuration file. All the properties listed below need to be present in userauthentication.properties file for the User Replication Login Servlet to work successfully. These properties are also shown with default assigned values.

```

# User Replication Debug
fuego.portal.containerAuthentication.userReplication.debug=true

# Fuego Directory Service Bindings to create new user.
fuego.portal.containerAuthentication.userReplication.dirServiceUser=root
fuego.portal.containerAuthentication.userReplication.dirServicePassword=password

# Fuego default roles to assign to new user. These roles must exist in Fuego's Directory
Service.
fuego.portal.containerAuthentication.userReplication.roles=R1;Run

# LDAP Bindings
fuego.portal.containerAuthentication.userReplication.additionalInfoFromLDAP=true

```

**fuego.portal.containerAuthentication.userReplication.debug:** This property can be assigned the literals “true” or “false”. If assigned the literal “true”, the User Replication Servlet will print out debug or trace messages into the Servlet Container Standard Output log file.

**fuego.portal.containerAuthentication.userReplication.dirServiceUser:** This property specifies an existing user in Fuego’s Directory Service with enough permissions to create new users.

**fuego.portal.containerAuthentication.userReplication.dirServicePassword:** This property specifies the password for the user specified in the above property.

**fuego.portal.containerAuthentication.userReplication.roles:** This property specifies the roles to be automatically granted to the new created user. The value assigned to this property is a “;” separated list of literals which represent the role names in the Fuego’s Directory Service. Each one of the referenced roles need to exist in Fuego’s Directory Service.

**fuego.portal.containerAuthentication.userReplication.additionalInfoFromLDAP:** This property can be assigned the literals “true” or “false”. If set to “false”, the user will be created with default values. No special customization will be performed to the user. A new user with the UserId coming from the Container authentication will be created with the default password “password”. No values will be assigned to the attributes “First Name”, “Last Name” and “Email”. If set to true, it will invoke a Java Class whose behavior is explained below.

If you want to replicate the user with more than just the UserId, you need to assign the literal “true” to the fuego.portal.containerAuthentication.additionalInfoFromLDAP property.

```

fuego.portal.containerAuthentication.userReplication.userRetrievalDataClass=fuego.portal.
servlet.deploy.UserPrincipalLoginUserJNDIRetrievalData

```

```
fuego.portal.containerAuthentication.userReplication.ldapConnectionFactory=com.sun.jndi.
ldap.LdapCtxFactory
fuego.portal.containerAuthentication.userReplication.ldapURL=ldap://192.168.0.135:389/dc
=victorysupport,dc=com
fuego.portal.containerAuthentication.userReplication.ldapConnectionUser=pvictory@victor
ysupport.com
fuego.portal.containerAuthentication.userReplication.ldapConnectionPassword=pvictory
fuego.portal.containerAuthentication.userReplication.ldapAuthentication=simple
fuego.portal.containerAuthentication.userReplication.ldapReferrals=ignore

# LDAP Implementation User Attributes Names to retrieve
fuego.portal.containerAuthentication.userReplication.userIdentifierAttribute=sAMAccount
Name
fuego.portal.containerAuthentication.userReplication.firstNameIdentifierAttribute=givenNa
me
fuego.portal.containerAuthentication.userReplication.lastNameIdentifierAttribute=sn
fuego.portal.containerAuthentication.userReplication.emailIdentifierAttribute=userPrincipa
lName
```

**fuego.portal.containerAuthentication.userReplication.userRetrievalDataClass:** This property specifies the name of the class to be invoked to retrieve additional information for the user being created. The name of the class needs to be package qualified. Furthermore, the following class needs to implement the interface “UserPrincipalLoginUserRetrievalData”.

The following properties are specific to the JNDI implementation of the “UserPrincipalLoginUserRetrievalData” interface with name “fuego.portal.servlet.deploy.UserPrincipalLoginUserJNDIRetrievalData”.

**fuego.portal.containerAuthentication.userReplication.ldapConnectionFactory:** This property specifies the java class name of the LDAP Connection Factory used to connect the LDAP Directory Service to get the extra User information. Sun’s JNDI LDAP Context Factory is the most widely used.

**fuego.portal.containerAuthentication.userReplication.ldapURL:** This property specified the LDAP Directory Service URL. This will may vary depending wihat LDAP Directory Service you are connecting to get the additional user information.

**fuego.portal.containerAuthentication.userReplication.ldapConnectionUser:** This property specifies the value of the user or principal used to connect the LDAP Directory Service.

**fuego.portal.containerAuthentication.userReplication.ldapConnectionPassword:** This property specifies the password or credentials for the user specified in the property above.

**fuego.portal.containerAuthentication.userReplication.ldapAuthentication:** This property specifies what kind of authentication is to be used when connecting to the LDAP Directory Service. By default you should use “simple”.

**fuego.portal.containerAuthentication.userReplication.ldapReferrals:** This property specifies how referrals should be managed by the connected LDAP Directory Service.

**fuego.portal.containerAuthentication.userReplication.userIdentifierAttribute:** This property specifies the name of the attribute in the connected LDAP Directory Service to find the created user.

**fuego.portal.containerAuthentication.userReplication.firstNameIdentifierAttribute:** This property specifies the name of User object attribute that contains the First name for the User being created and replicated.

**fuego.portal.containerAuthentication.userReplication.lastNameIdentifierAttribute:** This property specifies the name of User object attribute that contains the Last name for the User being created and replicated.

**fuego.portal.containerAuthentication.userReplication.emailIdentifierAttribute:** This property specifies the name of User object attribute that contains the email address for the User being created and replicated.

## Troubleshooting

It is very convenient that when errors take place, the administrator checks the Web Server (in this case Tomcat) log files. Most of the exceptions and errors are better described in these logs.

1. **Problem description:** Microsoft Active Directory has been configured as a forest of nodes and the user specified for connecting does not have enough permission to continue searching on the other forest Active Directory nodes.

```
2004-06-22 18:50:19 JNDIRealm[Standalone]: Connecting to URL
ldap://192.168.1.140/DC=scc,DC=labs,DC=fuegotech,DC=com
2004-06-22 18:50:19 JNDIRealm[Standalone]: Searching for Administrator
2004-06-22 18:50:19 JNDIRealm[Standalone]: base: filter: (sAMAccountName=Administrator)
2004-06-22 18:50:20 JNDIRealm[Standalone]: Exception performing authentication
javax.naming.PartialResultException: Unprocessed Continuation Reference(s); remaining name ''
    at com.sun.jndi.ldap.LdapCtx.processReturnCode(Unknown Source)
    at com.sun.jndi.ldap.LdapCtx.processReturnCode(Unknown Source)
    at com.sun.jndi.ldap.LdapNamingEnumeration.getNextBatch(Unknown Source)
    at com.sun.jndi.ldap.LdapNamingEnumeration.hasMoreImpl(Unknown Source)
    at com.sun.jndi.ldap.LdapNamingEnumeration.hasMore(Unknown Source)
    at org.apache.catalina.realm.JNDIRealm.getUserBySearch(JNDIRealm.java:1108)
    at org.apache.catalina.realm.JNDIRealm.getUser(JNDIRealm.java:985)
```

**Resolution:** The administrator needs to specify the value “follow” for the JNDI java.naming.referral property. This property can be specified in Tomcat’s JNDI Realm definition (in Tomcat’s server.xml) by setting “follow” the “referral” attribute.

2. **Problem description:** Microsoft Active Directory requires that the Domain name resolves to the host where the Microsoft Active Directory is deployed. An exception similar to this should be found in Tomcat’s log files.

```
2004-06-22 18:54:17 JNDIRealm[Standalone]: Connecting to URL
ldap://192.168.1.140/DC=scc,DC=labs,DC=fuegotech,DC=com
2004-06-22 18:54:18 JNDIRealm[Standalone]: Searching for Administrator
2004-06-22 18:54:18 JNDIRealm[Standalone]: base: filter: (sAMAccountName=Administrator)
2004-06-22 18:54:18 JNDIRealm[Standalone]: Exception performing authentication
javax.naming.PartialResultException [Root exception is javax.naming.CommunicationException:
scc.labs.fuegotech.com:389 [
Root exception is java.net.UnknownHostException: scc.labs.fuegotech.com]]
    at com.sun.jndi.ldap.LdapNamingEnumeration.hasMoreImpl(Unknown Source)
    at com.sun.jndi.ldap.LdapNamingEnumeration.hasMore(Unknown Source)
    at org.apache.catalina.realm.JNDIRealm.getUserBySearch(JNDIRealm.java:1108)
    at org.apache.catalina.realm.JNDIRealm.getUser(JNDIRealm.java:985)
```

**Resolution:** The administrator will need to make sure that both the host and domain return the same IP Address. You can verify this using the ping command. In our case, the Windows Domain “scc.labs.fuegotech.com” should return the IP Address 192.168.1.140. If this is a problem and your DNS cannot resolve the domain name to the same IP as the host where Microsoft Active Directory is deployed, you can edit the “hosts” file and enter the domain to map to this IP Address.

3. **Problem description:** Microsoft Active Directory is not being able to follow referrals.

```
2005-02-21 15:03:44 JNDIRealm[Standalone]: Connecting to URL
ldap://scc.labs.fuegotech.com:389/DC=fuego
2005-02-21 15:04:28 JNDIRealm[Standalone]: Searching for eduardoc
2005-02-21 15:04:28 JNDIRealm[Standalone]: base: filter:
(sAMAccountName=eduardoc.)
2005-02-21 15:04:49 JNDIRealm[Standalone]: Exception performing
authentication
javax.naming.PartialResultException [Root exception is
javax.naming.CommunicationException: scc.labs.fuegotech.com:389 [Root
exception is
java.net.ConnectException: Connection timed out: connect]]
    at com.sun.jndi.ldap.LdapNamingEnumeration.hasMoreImpl(Unknown Source)
    at com.sun.jndi.ldap.LdapNamingEnumeration.hasMoreReferrals(Unknown
Source)
    at com.sun.jndi.ldap.LdapNamingEnumeration.hasMoreImpl(Unknown Source)
    at com.sun.jndi.ldap.LdapNamingEnumeration.hasMoreReferrals(Unknown
Source)
    at com.sun.jndi.ldap.LdapNamingEnumeration.hasMoreImpl(Unknown Source)
    at com.sun.jndi.ldap.LdapNamingEnumeration.hasMore(Unknown Source)
    at
org.apache.catalina.realm.JNDIRealm.getUserBySearch(JNDIRealm.java:1109)
...
Caused by: javax.naming.CommunicationException: scc.labs.fuegotech.com:389
[Root exception is
java.net.ConnectException: Connection timed out: connect]
    at com.sun.jndi.ldap.LdapReferralContext.<init>(Unknown Source)
    at com.sun.jndi.ldap.LdapReferralException.getReferralContext(Unknown
Source)
    at com.sun.jndi.ldap.LdapNamingEnumeration.hasMoreReferrals(Unknown
Source)
    ... 34 more
Caused by: java.net.ConnectException: Connection timed out: connect
    at java.net.PlainSocketImpl.socketConnect(Native Method)
    at java.net.PlainSocketImpl.doConnect(Unknown Source)
    at java.net.PlainSocketImpl.connectToAddress(Unknown Source)
    at java.net.PlainSocketImpl.connect(Unknown Source)
    at java.net.Socket.connect(Unknown Source)
    at java.net.Socket.connect(Unknown Source)
```

**Resolution:** The administrator needs to specify the value “ignore” for the JNDI java.naming.referral property. This property can be specified in Tomcat’s JNDI Realm definition (in Tomcat’s server.xml) by setting “ignore” the “referral” attribute. This is a safe solution if all the information that needs to be collected for user authentication resides on the main node specified in the connection LDAP URL.

4. **Problem description:** Cannot authentication the user through the Fuego Portal login page due to incorrect specification of the Role in the web.xml. An error similar to this is returned in the browser.

```
HTTP Status 403 - Access to the requested resource has been denied
```

**Resolution:** Make sure the “role-name” attribute within the “auth-constraint” tag in the Fuego Work Portal web.xml is a valid role. In some cases, the LDAP URL is case sensitive so the administrator should pay attention to this fact.





## Appendix A

This appendix provides the description of the Java Interfaces to be implemented if choosing a different source to retrieve additional information when replicating a user into Fuego's Directory Service. It also provides the abstract class returned by the UserPrincipalLoginUserRetrievalData interface method to be used by the UserPrincipalLoginWithUserAuthentication servlet. This class needs to be extended to include any particularities for the new User Retrieval Data interface. If no specific details are needed it needs to at least extended for the class to be instantiated from the class implementing the UserPrincipalLoginUserRetrievalData interface.

```
package fuego.portal.servlet.deploy;

import java.util.Hashtable;

public interface UserPrincipalLoginUserRetrievalData
{
    public UserPrincipalLoginUserData getUserRelatedData(String principal, Hashtable
authenticationProperties);
}
```

```
package fuego.portal.servlet.deploy;

public abstract class UserPrincipalLoginUserData
{
    //~ Instance fields .....

    private String displayName_d;
    private String emailAddress_d;
    private String faxNbr_d;
    private String firstName_d;
    private String lastName_d;
    private String password_d;
    private String phoneNbr_d;

    //~ Constructors .....

    public UserPrincipalLoginUserData() { }

    //~ Methods .....

    public void setDisplayName(String dispName)
    {
```

```
        displayName_d = dispName;
    }

    public String getDisplayName()
    {
        return displayName_d;
    }

    public void setEmailAddress(String email)
    {
        emailAddress_d = email;
    }

    public String getEmailAddress()
    {
        return emailAddress_d;
    }

    public void setFaxNumber(String faxNbr)
    {
        faxNbr_d = faxNbr;
    }

    public String getFaxNumber()
    {
        return faxNbr_d;
    }

    public void setFirstName(String firstName)
    {
        firstName_d = firstName;
    }

    public String getFirstName()
    {
        return firstName_d;
    }

    public void setLastName(String lastName)
    {
        lastName_d = lastName;
    }

    public String getLastName()
    {
        return lastName_d;
    }
}
```

```

    }

    public void setPassword(String password)
    {
        password_d = password;
    }

    public String getPassword()
    {
        return password_d;
    }

    public void setPhoneNumber(String phoneNbr)
    {
        phoneNbr_d = phoneNbr;
    }

    public String getPhoneNumber()
    {
        return phoneNbr_d;
    }

    public String toString()
    {
        return "LastName: (" + lastName_d + ")\nFirstName: (" + firstName_d +
        ")\nDisplay Name: (" +
            displayName_d + ")\nEmail: (" + emailAddress_d + ")\nPhone Nbr: (" +
        phoneNbr_d +
            ")\nFax Number: (" + faxNbr_d + ")";
    }
}

```