



# Define BA Server Advanced Security

## Copyright Page

---

This document supports Pentaho Business Analytics Suite 5.3 GA and Pentaho Data Integration 5.3 GA, documentation revision January 15th, 2015, copyright © 2015 Pentaho Corporation. No part may be reprinted without written permission from Pentaho Corporation. All trademarks are the property of their respective owners.

### Help and Support Resources

To view the most up-to-date help content, visit <https://help.pentaho.com>.

If you do not find answers to your questions here, please contact your Pentaho technical support representative.

Support-related questions should be submitted through the Pentaho Customer Support Portal at <http://support.pentaho.com>.

For information about how to purchase support or enable an additional named support contact, please contact your sales representative, or send an email to [sales@pentaho.com](mailto:sales@pentaho.com).

For information about instructor-led training, visit <http://www.pentaho.com/training>.

### Liability Limits and Warranty Disclaimer

The author(s) of this document have used their best efforts in preparing the content and the programs contained in it. These efforts include the development, research, and testing of the theories and programs to determine their effectiveness. The author and publisher make no warranty of any kind, express or implied, with regard to these programs or the documentation contained in this book.

The author(s) and Pentaho shall not be liable in the event of incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or use of the programs, associated instructions, and/or claims.

### Trademarks

The trademarks, logos, and service marks ("Marks") displayed on this website are the property of Pentaho Corporation or third party owners of such Marks. You are not permitted to use, copy, or imitate the Mark, in whole or in part, without the prior written consent of Pentaho Corporation or such third party. Trademarks of Pentaho Corporation include, but are not limited, to "Pentaho", its products, services and the Pentaho logo.

Trademarked names may appear throughout this website. Rather than list the names and entities that own the trademarks or inserting a trademark symbol with each mention of the trademarked name, Pentaho Corporation states that it is using the names for editorial purposes only and to the benefit of the trademark owner, with no intention of infringing upon that trademark.

### Third-Party Open Source Software

For a listing of open source software used by each Pentaho component, navigate to the folder that contains the Pentaho component. Within that folder, locate a folder named licenses. The licenses folder contains HTML files that list the names of open source software, their licenses, and required attributions.

### Contact Us

Global Headquarters Pentaho Corporation Citadel International, Suite 460

5950 Hazeltine National Drive Orlando, FL 32822

Phone: +1 407 812-OPEN (6736)

Fax: +1 407 517-4575

<http://www.pentaho.com>

Sales Inquiries: [sales@pentaho.com](mailto:sales@pentaho.com)

## Introduction

---

Your predefined users and roles can be used if you are already using a security provider such as LDAP, Microsoft Active Directory, or Single Sign-On. These articles guide you through the process of configuring third-party security frameworks for the Pentaho BA Server.

If you are evaluating Pentaho or have a production environment with fewer than a hundred users, you may decide to use Pentaho [default security](#).

## Prerequisites

Before you can implement advanced security, you must have [installed](#) and [configured](#) the BA Server. If you chose to [install the DI server](#) and its design tool, there is a separate section for [configuring them](#).

## Expertise

The topics within this series of articles are written for security administrators with knowledge of the security provider to be used, details about their user community and a plan for which roles to use in the Pentaho system, and how to use the command line to issue commands for Microsoft Windows or Linux.

## Tools

We provide a web application, the User Console, which you use to perform most security tasks. Some of these security tasks require that you work on the actual machine that has the BA software installed.

## Login Credentials

All of the tasks that use the User Console, **Administration** page, require that you [log on to the User Console](#) with the Pentaho administrator user name and password.

## Related Articles

These articles explain how to administer, fine-tune, and troubleshoot Pentaho systems.

For BA only:

- [Administer BA Server](#)
- [Optimize BA Server Performance](#)
- [Troubleshoot BA Server Issues](#)

For DI only:

- [Administer DI Server](#)

- [Define DI Server Advanced Security](#)
- [Troubleshoot DI Server Issues](#)

## Security Overview

We support two different security options: Pentaho Security or advanced security providers, such as LDAP, Single Sign-On, or Microsoft Active Directory. This table can help you choose the option that is best for your environment.

Table 1. Security Decision Table

| Explore Considerations | Choose Options   |  |
|------------------------|--|--|
|                        | <a href="#">Pentaho Security</a>   | <a href="#">Advanced Security Providers—LDAP, Single Sign-On, or Microsoft Active Directory</a>  |
| Summary                | <p>Pentaho Security is the easiest way to configure security quickly. The User Console enables you to define and manage users and roles. The BA Server controls which users and roles can access web resources through the User Console or resources in the Pentaho BA repository.</p> <p>Pentaho Security works well if you do not have a security provider or if you have a user community with less than 100 users.</p> | <p>If you are already using a security provider, such as LDAP, Single Sign-On, or Microsoft Active Directory, you can use the users and roles you have already defined with Pentaho. Your security provider controls which users and roles can access Pentaho web resources through the User Console or resources in the BA repository.</p> <p>Advanced security scales well for production and enterprise user communities.</p> |
| Expertise              | <p>Knowledge of your user community and which users should have which roles in the Pentaho system.</p> <p>Knowledge about security in general is <i>not</i> required.</p>  | <p>Knowledge of your user community and which users should have which roles in the Pentaho system.</p> <p>Knowledge about your particular security provider and its options is required.</p>   |
| Time                   | It takes approximately 5 minutes per user and role to configure Pentaho Security.  | It takes approximately 1 hour to configure the BA Server to use your existing security provider.   |
| Recommendation         | Recommended for the Pentaho Trial Download, evaluating, and rapid development.   | Recommended for production.  |

## Implement Advanced Security

---

This section discusses several different ways to handle Security other than with default Pentaho security.

- [Switch to MS Active Directory](#)
- [Switch to LDAP](#)
- [Manual MSAD Configuration](#)
- [Manual LDAP Configuration](#)
- [Manual JDBC Connection Configuration](#)
- [Manual LDAP/JDBC Hybrid Configuration](#)
- [Use Single Sign-On](#)
- [Add Web Resource Authentication](#)

## Switch to MS Active Directory

---

1. From User Console **Home** menu, click **Administration**, then select **Authentication** from the left. The **Authentication** interface appears. **Local - Use basic Pentaho Authentication** is selected by default.
2. Choose the **External - Use LDAP / Active Directory server** radio button. The **LDAP Server Connection** fields populate with a default URL, user name, and password.
3. Change the **Server URL**, **User Name**, and **Password** as needed.
4. Click **Test Server Connection** to verify the connection to your server and to complete the set up.
5. Click the Browse buttons to select the **Pentaho System Administrator** user and role to match your configuration. Click **OK**. The text box auto-populates with the selected values.
6. For MSAD, choose **Custom Configuration**.
7. For **Users**:

- a. **Search Base** by entering the path where your users are located. Example:

```
CN=Users,DC=MyDomain,DC=com
```

- b. **Search Filter** by entering in the attribute that users will login with. Example:

```
(sAMAccountName={0})
```

8. For **Roles**:

- a. For **Role Attributes**, enter in the Attribute that is used for roles/groups. Example:

```
CN
```

- b. For a **Role Search Filter**, enter in the ObjectClass that defines that these are roles or groups. Example:

```
(&(objectClass=group)(CN=Pentaho*))
```

- c. For **Role Search Base**, enter in the path where your roles or groups are located. Example:

```
OU=groups,DC=MyDomain,DC=com
```

- a. Click **Test**.

9. For **Populator**:

- a. For **Group Role Attribute**, enter in the Attribute that is used for groups. Example:

```
CN
```

- b. For **Group Search Base**, enter in the path to where your groups are located. Example:

```
OU=groups,DC=MyDomain,DC=com
```

- c. Set the **Group Search Filter** to

```
(member:1.2.840.113556.1.4.1941:={0})
```

You can set a **Role Prefix** if you need one to filter by.

10. Click **Test**, then click **Save**.
11. Shut down the BA Server.
12. Locate these three files and modify the settings as noted.
  - a. Navigate to the `pentaho-solutions/system` directory, and open the **repository.spring.properties** file with a text editor. Find these two sections and edit them to match your Active Directory settings, then save and close the file.

```
singleTenantAdminUserName=admin
```

```
singleTenantAdminAuthorityName=Administrator
```

- b. In the `pentaho-solutions/system` directory, open the `pentaho.xml` file with a text editor. Find this section and edit it to match your Active Directory settings, then save and close the file.

```
<acl-voter> <admin-role>Administrator</admin-role> </acl-voter>
```

- c. Navigate to the `pentaho-solutions/system/data-access` directory, and open the **settings.xml** file with a text editor. Find these two sections and edit them to match your Active Directory settings, then save and close the file.

```
<data-access-roles>Administrator</data-access-roles>
```

```
<data-access-view-roles>Authenticated,Administrator</data-access-view-roles>
```

13. Restart the BA Server.

The BA Server is now configured to authenticate users against your MSAD server.



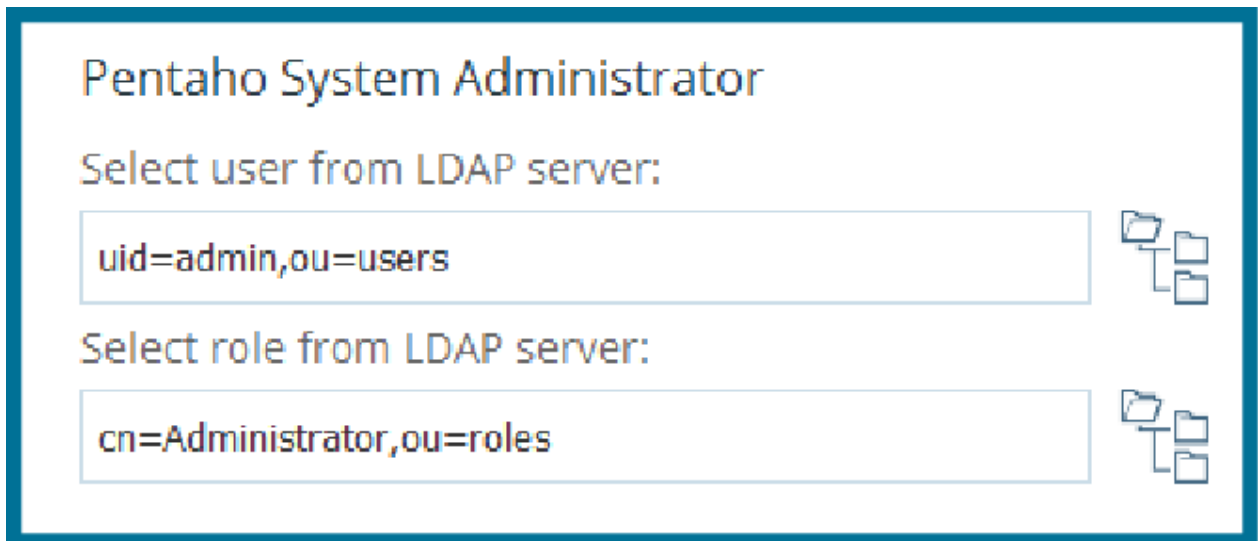
## Switch to LDAP

---

Note: In order to use SSL to connect to your LDAP server, you will need to import the certificate into the JRE's trustStore/keyStore used to by the BA Server (java/lib/security/cacerts).

1. From User Console **Home** menu, click **Administration**, then select **Authentication** from the left. The **Authentication** interface appears. **Local - Use basic Pentaho Authentication** is selected by default.
2. Choose the **External - Use LDAP / Active Directory server** radio button. The **LDAP Server Connection** fields populate with a default URL, user name, and password.
3. Change the **Server URL**, **User Name**, and **Password** as needed.
4. Click **Test Server Connection** to verify the connection to your LDAP server and to complete the set up.
5. Click the node to select the **Pentaho System Administrator** user and role to match your LDAP configuration. Click **OK**.

Note: The Admin user is required for all system-related operations, including the creation of user folders. The Administrator Role is required for mapping a third-party admin role to the Pentaho admin role (Administrator). This is required for all ABS functionality to work properly.



**Pentaho System Administrator**

Select user from LDAP server:

uid=admin,ou=users

Select role from LDAP server:

cn=Administrator,ou=roles

6. Choose your **LDAP Provider** from the drop-down menu.
7. Configure the LDAP connection as explained in [LDAP Properties](#). Click **Test**.

The BA Server is now configured to authenticate users against your LDAP directory server.

## Manual MSAD Configuration

---

The server does not recognize any difference among LDAP-based directory servers, including Active Directory. However, the way that you modify certain LDAP-specific files will probably be different for Microsoft Active Directory (MSAD) than for more traditional LDAP implementations. Below are some tips for specific MSAD-specific configurations that you might find helpful.

### Binding

MSAD allows you to uniquely specify users in two ways, in addition to the standard DN. If the standard DN is not working, try one of the two below. Each of the following examples is shown in the context of the `userDn` property of the Spring Security `DefaultSpringSecurityContextSource` bean.

Note: The examples in this section use `DefaultSpringSecurityContextSource`. Be aware that you may need to use the same notation (Kerberos or Windows domain) in all of your DN patterns.

Kerberos notation example for [pentahoadmin@mycompany.com](mailto:pentahoadmin@mycompany.com):

File: `applicationContext-security-ldap.properties`

```
contextSource.providerUrl=ldap://mycompany\:389
contextSource.userDn=pentahoadmin@mycompany.com
contextSource.password=omitted
```

Windows domain notation example for `MYCOMPANY\pentahoadmin`:

File: `applicationContext-security-ldap.properties`

```
contextSource.providerUrl=ldap://mycompany\:389
contextSource.userDn=MYCOMPANY\pentahoadmin
contextSource.password=omitted
```

### Referrals

If more than one Active Directory instance is serving directory information, it may be necessary to enable referral following. This is accomplished by modifying the `DefaultSpringSecurityContextSource` bean.

```
<bean id="contextSource" class="org.springframework.security.ldap.
DefaultSpringSecurityContextSource">
  <constructor-arg value="${contextSource.providerUrl}"/>
  <property name="userDn" value="${contextSource.userDn}"/>
```

```
<property name="password" value="{contextSource.password}"/>
<property name="referral" value="follow" />
</bean>
```

## User DN Patterns vs. User Searches

In the **LdapAuthenticator** implementations provided by Spring Security (**BindAuthenticator** for instance), you must either specify a **userDnPatterns**, or a **userSearch**, or both. If you're using the Kerberos or Windows domain notation, you should use **userDnPatterns** exclusively in your **LdapAuthenticator**.

Note: The reason for suggesting **userDnPatterns** when using Kerberos or Windows domain notation is that the **LdapUserSearch** implementations do not give the control over the DN that **userDnPatterns** does. (The **LdapUserSearch** implementations try to derive the DN in the standard format, which might not work in Active Directory.)

Note, however, that **LdapUserDetailsService** requires an **LdapUserSearch** for its constructor.

User DN Pattern example:

```
<bean id="authenticator"
class="org.springframework.security.providers.ldap.authenticator.
BindAuthenticator">
<constructor-arg>
    <ref local="contextSource"/>
</constructor-arg>
<propertyname="userDnPatterns">
<list>
    <value>{0}@mycompany.com
</value> <!-- and/or -->
    <value>domain\{0}</value>
</list>
</property>
</bean>
```

In user searches, the **sAMAccountName** attribute should be used as the user name. The **searchSubtree** property (which influences the **SearchControls**) should most likely be true. Otherwise, it searches the specified base plus one level down.

User Search example:

```
<bean id="userSearch"
class="org.springframework.security.ldap.search.FilterBasedLdapUserSearch">
    <constructor-arg index="0" value="DC=mycompany,DC=com" />
    <constructor-arg index="1">
        <value>(sAMAccountName={0})</value>
    </constructor-arg>
</bean>
```

```
</constructor-arg> <constructor-arg index="2">
<ref local="contextSource" />
</constructor-arg>
<property name="searchSubtree" value="true"/>
</bean>
```

## Nested Groups

You can remove nested or transitive groups out of Active Directory. In the LDAP popular group filter, enter the following LDAP filter for MSAD nested groups:

```
(member:1.2.840.113556.1.4.1941:={0})
```

This will search down the whole tree of nested groups.

## Manual LDAP Configuration

---

You must have a working directory server with an established configuration before continuing.

Follow the instructions below to manually switch from Pentaho default security to LDAP security.

1. Stop the BA Server and User Console.
2. Change the `securities.properties` file located at `/pentaho-solutions/system` folder from `provider=jackrabbit` to `provider=ldap`.
3. Save and close the file, then edit the `/pentaho-solutions/system/applicationContext-security-ldap.properties` file and modify the **localhost** and **password** to match your configuration.

```
contextSource.providerUrl=ldap://localhost:10389/ou\=system
```

```
contextSource.password=secret
```

4. Save and close the file, then edit the `/pentaho-solutions/system/data-access/settings.xml` file and modify the settings to match your LDAP configuration. Find and replace the entries for **Administrator** in the examples below with the correct administrator name for your LDAP configuration.

```
<!-- roles with data access permissions -->
<data-access-roles>Administrator</data-access-roles>
<!-- users with data access permissions -->
<!--
<data-access-users></data-access-users>
-->
<!-- roles with datasource view permissions -->
<data-access-view-roles>Authenticated,Administrator</data-
access-view-roles>
<!-- users with datasource view permissions -->
<!-- <data-access-view-users>suzy</data-access-view-users> -->
<!-- default view acs for user or role -->
<data-access-default-view-acls>31</data-access-default-view-
acsl>
```

5. Save and close the file, then edit the following files in the `/pentaho/server/biserver-ee/pentaho-solutions/system/` directory and change all instances of the **Administrator** and **Authenticated** role values to match the appropriate roles in your LDAP configuration:
  - `pentaho.xml`
  - `repository.spring.properties`
  - `applicationContext-spring-security.xml`
9. Delete these two folders from the `/pentaho/server/biserver-ee/pentaho-solutions/system/jackrabbit/repository` directory:
  - `repository`
  - `workspaces`
12. Restart the BA Server and test the LDAP functionality.

The BA Server is now configured to authenticate users against your directory server. The [LDAP Properties](#) reference article contains supplemental information for LDAP values.

- [Use Nested Roles in LDAP](#)
- [LDAP Properties](#)

## Use Nested Roles in LDAP

---

It is possible to nest user roles such that one role includes all of the users of another role. Doing this external to the core LDAP structure prevents recursive directory queries to find all parents of a given child role. Follow the directions below to modify the BA Server to support nested roles for LDAP and MSAD authentication types.

1. Stop the BA Server or service.

```
sh /usr/local/pentaho/server/biserver-ee/stop-pentaho.sh
```

2. Open the `/pentaho/server/biserver-ee/pentaho-solutions/system/applicationContext-spring-security-ldap.xml` file with a text editor.
3. In the **populator** bean definition, replace **DefaultLdapAuthoritiesPopulator** with **NestedLdapAuthoritiesPopulator**

```
<bean id="populator" class="org.pentaho.platform.plugin.services.security.userrole.ldap.NestedLdapAuthoritiesPopulator">
```

4. Save the file, then edit `/pentaho/server/biserver-ee/pentaho-solutions/system/applicationContext-pentaho-security-ldap.xml`. This and the next step are only necessary if the roles that serve as "parents" to nested roles cannot be returned by a traditional all authorities search.
5. Add an **extraRoles** bean to the list of transformers in the **ChainedTransformers** bean, and set properties for each parent role (represented by example\_role below).

```
<bean id="allAuthoritiesSearch" class="org.pentaho.platform.plugin.services.security.userrole.ldap.search.GenericLdapSearch">
  <!-- omitted -->
  <constructor-arg index="2">
    <bean class="org.apache.commons.collections.functors.ChainedTransformer">
      <constructor-arg index="0">
        <list>
          <bean class="org.pentaho.platform.plugin.services.security.userrole.ldap.transform.SearchResultToAttrValueList">
            <!-- omitted -->
          </bean>
          <bean class="org.pentaho.platform.plugin.services.security.userrole.ldap.transform.ExtraRoles">
            <property name="extraRoles">
              <set>
```

```
        <value>example_role</value>
      </set>
    </property>
  </bean>
  <bean class="org.pentaho.platform.plugin.services.
security.userrole.ldap.transform.StringToGrantedAuthority">
    <!-- omitted -->
  </bean>
</list>
</constructor-arg>
</bean>
</constructor-arg>
</bean>
```

6. Save the file, close your text editor, and start the BA Server.

```
sh /usr/local/pentaho/server/biserver-ee/start-pentaho.sh
```

The BA Server can now efficiently handle nested roles with LDAP or Active Directory authentication.



## LDAP Properties

You can configure LDAP values by editing the `/pentaho-solutions/system/applicationContext-security-ldap.properties` file in your BA Server or DI Server directory, or through the User Console for the BA Server (the LDAP options in the console apply only to the BA Server, not the DI Server).

### Connection Information (Context)

These entries define connections involving LDAP users (typically administrators) that can execute directory searches.

| LDAP Property                          | Purpose  | Example   |
|--|--|---|
| <code>contextSource.providerUrl</code> | LDAP connection URL  | <code>contextSource.providerUrl=ldap://holly:389/DC=Valyant,DC=local</code>     |
| <code>contextSource.userDn</code>      | Distinguished name of a user with read access to directory | <code>contextSource.userDn=CN=Administrator,CN=Users,DC=Valyant,DC=local</code> |
| <code>contextSource.password</code>    | Password for the specified user                            | <code>contextSource.password=secret</code>                                      |

### Users

These options control how the LDAP server is searched for user names that are entered in the Pentaho login dialog box.

Note: The `{0}` token will be replaced by the user name from the login dialogue.

Note: The example above defines `DC=Valyant,DC=local` in `contextSource.providerURL`. Given that definition, you would not need to repeat that in `userSearch.searchBase` below because it will be appended automatically to the defined value here.

| LDAP Property                        | Purpose  | Example   |
|--------------------------------------|--|---|
| <code>userSearch.searchBase</code>   | Base (by user name) for user searches  | <code>userSearch.searchBase=CN=Users</code>               |
| <code>userSearch.searchFilter</code> | Filter (by user name) for user searches. The attribute you specify here must contain the value that you want your users to log into Pentaho with. Active Directory user names are represented by <code>sAMAccountName</code> ; | <code>userSearch.searchFilter=(sAMAccountName={0})</code> |

| LDAP Property | Purpose  | Example |
|---------------|--|---------|
|               | full names are represented by <b>displayName</b> . |         |

## Populator

The populator matches fully distinguished user names from **userSearch** to distinguished role names for roles those users belong to.

Note: The {0} token will be replaced with the user DN found during a user search; the {1} token is replaced with the user name entered in the login screen.

| LDAP Property                | Purpose   | Example  |
|------------------------------|---|--|
| populator.convertToUpperCase | Indicates whether or not retrieved role names are converted to uppercase  | populator.convertToUpperCase=false                                   |
| populator.groupRoleAttribute | The attribute to get role names from  | populator.groupRoleAttribute=cn                                      |
| populator.groupSearchBase    | Base (by user DN or user name) for role searches.   | populator.groupSearchBase=ou=Pentaho                                 |
| populator.groupSearchFilter  | The special nested group filter for Active Directory is shown in the example; this will not work with non-MSAD directory servers.                               | populator.groupSearchFilter=(memberof:1.2.840.113556.1.4.1941:={0})) |
| populator.rolePrefix         | A prefix to add to the beginning of the role name found in the group role attribute; the value can be an empty string.  | populator.rolePrefix=  |
| populator.searchSubtree      | Indicates whether or not the search must include the current object and all children. If set to <b>false</b> , the search must include the current object only. | populator.searchSubtree=true   |

## All Authorities Search

These entries populate the BA Server Access Control List (ACL) roles. These should be similar or identical to the Populator entries.

| LDAP Property                      | Purpose                            | Example                                    |
|------------------------------------|------------------------------------|--|
| allAuthoritiesSearch.roleAttribute | The attribute used for role values | allAuthoritiesSearch.roleAttribute=cn      |
| allAuthoritiesSearch.searchBase    | Base for "all roles" searches      | allAuthoritiesSearch.searchBase=ou=Pentaho |

| LDAP Property                     | Purpose   | Example   |
|-----------------------------------|---|---|
| allAuthoritiesSearch.searchFilter | Filter for "all roles" searches. Active Directory requires that the <b>objectClass</b> value be set to <b>group</b> . | allAuthoritiesSearch.searchFilter=(objectClass=group) |

## All user name search

These entries populate the BA Server ACL users.

| LDAP Property                         | Purpose                            | Example   |
|---------------------------------------|------------------------------------|---|
| allUsernamesSearch.username Attribute | The attribute used for user values | allUsernamesSearch.username Attribute= sAMAccountName |
| allUsernamesSearch.searchBase         | Base for "all users" searches      | allUsernamesSearch.searchBase= CN=users               |
| allUsernamesSearch.searchFilter       | Filter for "all users" searches    | allUsernamesSearch.searchFilter= objectClass=person   |

## Manual JDBC Connection Configuration

---

You must have existing security tables in a relational database in order to proceed with this task.

Follow the instructions below to switch from Pentaho default security to JDBC security, which will allow you to use your own security tables.

Note: If you are using the BA Server and choose to switch to a JDBC security shared object, you will no longer be able to use the role and user administration settings in the Administration portion of the User Console.

1. Stop the BA Server by running the **stop-pentaho** script.
2. Open `/pentaho-solutions/system/security.properties` with a text editor.
3. Change the value of the `provide` property to `jdbc`.
4. Set up the connection to the database that holds the user/authorities.
  - a. Open the `/pentaho-solutions/system/applicationContext-spring-security-jdbc.properties` file with a text editor. Find these two lines and change the **jdbcDriver** and **URL** as appropriate.

```
datasource.driver.classname=org.hsqldb.jdbcDriver
```

```
datasource.url=jdbc:hsqldb:hsqldb://localhost:9002/userdb
```

- b. Change the user name and password by editing these two items.

```
\datasource.username=sa, datasource.password=
```

- c. Set the **validation query** by editing this row. There are examples of different validation queries in the file.

```
datasource.validation.query=SELECT 1 FROM INFORMATION_SCHEMA.SYSTEM_
USERS
```

- d. Set the **wait timeout**, **max pool**, and **max idle** by editing these three items to change the defaults.

```
datasource.pool.max.wait=-1, datasource.pool.max.active=8, datasource.
max.idle=4
```

- e. Save the file and close the editor.
5. If you need to, modify these two queries that pull information about users/authorities.
  - a. Open `/pentaho-solutions/system/applicationContext-spring-security-jdbc.xml` with a text editor.
  - b. Find this line and change the **query** that returns the user and roles that the user is a member of as appropriate.

```
<value>
    <![CDATA[SELECT username, authority FROM GRANTED_AUTHORITIES
WHERE username = ? ORDER BY authority]]>
</value>
```

- c. Find this line and change the **query** that determines the user, password, and whether they can log in as appropriate.

```
<value>
    <![CDATA[SELECT username, password, enabled FROM USERS WHERE
username = ? ORDER BY username]]>
</value>
```

6. If you need to, modify these three queries that pull information about users/authorities.

- a. Open the /pentaho-solutions/system/applicationContext-pentaho-security-jdbc.xml file with a text editor.
- b. Find this line and change the **query** that shows the roles for security on objects as appropriate.

```
<value>
    <![CDATA[SELECT distinct(authority) as authority FROM AUTHORITIES
ORDER BY authority]]>
</value>
```

- c. Find this line and change the **query** that returns all users in a specific role as appropriate.

```
<value>
    <![CDATA[SELECT distinct(username) as username FROM GRANTED_
AUTHORITIES where authority = ? ORDER BY username]]>
</value>
```

- d. Find this line and change the **query** that returns all users in a specific role as appropriate.

```
<value>
    <![CDATA[SELECT distinct(username) as username FROM USERS ORDER
BY username]]>
</value>
```

- e. Save the file and close the editor.

7. Update the default Pentaho admin user on the system to map to your JDBC admin user.

- a. Open the /pentaho-solutions/system/repository.spring.properties file with a text editor.
- b. Find these lines and change the default value from <admin> to map to your <admin username> in your JDBC system.

```
singleTenantAdminUserName=<Admin User>
```

- c. Save the file and close the editor.

8. To fully map the JDBC's admin role to other configuration files, specify the name of the administrator role for your JDBC authentication database in the `applicationContext-pentaho-security-jdbc.xml` file.

- a. Open the `/pentaho-solutions/system/applicationContext-pentaho-security-jdbc.xml` file with a text editor.
- b. Find these lines and change the entry key to the key assigned to the administrator role in your JDBC authentication database.

```
<!-- map ldap role to pentaho security role -->
<util:map id="jdbcRoleMap">
  <entry key="Admin" value="Administrator"/>
</util:map>
```

- c. Save and close the file.

9. Start the server by running the **start-pentaho** script.

The server is configured to authenticate users against the specified database.

## Manual LDAP/JDBC Hybrid Configuration

---

You must have a working directory server with an established configuration, and a database containing your user roles before continuing.

It is possible to use a directory server for user authentication and a JDBC security table for role definitions. This is common in situations where LDAP roles cannot be redefined for BA Server use. Follow the below instructions to switch the BA Server's authentication back end from the Pentaho data access object to an LDAP/JDBC hybrid.

Note: Replace the **pentahoAdmins** and **pentahoUsers** references in the examples below with the appropriate roles from your LDAP configuration.

1. Stop the BA Server and User Console.
2. Open `/pentaho-solutions/system/security.properties` with a text editor.
3. Add this value beneath the `provider=ldap` line, then save and close the file:

```
role.provider=jdbc
```

4. Open the `/pentaho-solutions/system/pentahoObjects.spring.xml` with a text editor.
5. Find these code blocks and change the **providerName** to `jdbc`.

```
<!-- Reference to a bean in one of the applicationContext-pentaho-security-*.
xml; selected by configured provider-->
<pen:bean id="activeUserRoleListService" class="org.pentaho.platform.api.
engine.IUserRoleListService">
  <pen:attributes>
    <pen:attr key="providerName" value="${security.provider}"/>
  </pen:attributes>
</pen:bean>
```

6. Open the `/pentaho-solutions/system/applicationContext-spring-security-jdbc.properties` file with a text editor and edit to show your database connection information. Save and close the file.
7. Open `/pentaho-solutions/system/applicationContext-pentaho-security-jdbc.xml`. Find this code block and change **Admin** to an appropriate administrator role in your JDBC authentication database.

```
<!-- map ldap role to pentaho security role -->
<util:map id="jdbcRoleMap">
  <entry key="Admin" value="Administrator"/>
</util:map>
```

8. Open the `/pentaho-solutions/system/applicationContext-springsecurity-ldap.xml` file and replace the populator bean definition with this one.

```
<bean id="populator" class="org.springframework.security.
ldap.populator.UserDetailsServiceLdapAuthoritiesPopulator">
<constructor-arg ref="jdbcUserDetailsService" />
</bean>
```

9. Delete the `/tomcat/work/` and `/tomcat/temp/` directories.

10. Start the BA Server and User Console.

11. Log into the User Console.

12. Configure the Pentaho LDAP connection as explained in [LDAP Properties](#).

The BA Server is configured to authenticate users against your directory server.



## Use Single Sign-On

---

This section contains instructions for configuring the BA Server to work with a single sign-on (SSO) framework. At this time, only Central Authentication Service (CAS) and Integrated Windows Authentication (IWA) are supported. Refer only to the instructions below that apply to the framework you are using.

- [Switch to Central Authentication Service \(CAS\)](#)
- [Switch to Integrated Windows Authentication \(IWA\)](#)

## Switch to Central Authentication Service (CAS)

---

Pentaho integrates with Central Authentication Service (CAS). You must have a CAS server installed and running before you continue.

1. Stop the BA Server.
2. Download the [cas-client-core-3.1.5.jar](#) and copy it to `biserver-ee/tomcat/webapps/pentaho/WEB-INF/lib` folder.
3. Download the [spring-security-cas-client-2.0.5.RELEASE.jar](#) and copy it to `biserver-ee/tomcat/webapps/pentaho/WEB-INF/lib` folder.
4. Open the `pentaho-spring-beans.xml` file with any file editor and update it as follows.
  - a. Add `<import resource="applicationContext-spring-security-cas.xml" />` to the list of imports after all other `applicationContext*.xml` files.
5. Open the `applicationContext-spring-security-cas.xml` file with any file editor and update it as follows.
  - a. Change all the references of this URL <https://localhost:8443/cas> to your working CAS server URL.
  - b. If you are using Pentaho with SSL, then update references to this URL: <http://localhost:8080/pentaho>.
  - c. Go into the file from the previous step and find the section for `casAuthenticationProvider.UserDetailsService`.

```
<bean id="casAuthenticationProvider"
class="org.springframework.security.providers.cas.
CasAuthenticationProvider">
<property name="userDetailsService">
  <ref bean="userDetailsService" />
</property>
```

Change it based on your configuration to the appropriate one as shown below.

```
casAuthenticationProvider.MemoryUserDetailsService
```

```
casAuthenticationProvider.hibernateUserDetailsService
```

```
casAuthenticationProvider.jdbcUserDetailsService
```

```
casAuthenticationProvider.ldapUserDetailsService
```

Note: You must use the publicly available IP address for all URLs in this file.

6. Add the following in their respective sections to the `web.xml`.

```
<servlet>
  <servlet-name>casFailed</servlet-name>
  <jsp-file>/jsp/casFailed.jsp</jsp-file>
</servlet>
```

```
<servlet-mapping>
  <servlet-name>casFailed</servlet-name>
  <url-pattern>/public/casFailed</url-pattern>
</servlet-mapping>
```

```
<!--
<listener>
  <listener-class>org.jasig.cas.client.session.
SingleSignOutHttpSessionListener</listener-class>
</listener>
-->
```

7. If you are using a self-signed certificate, you must do these steps. If not, follow the instructions in step 8.
  - a. For **memory** only, open the **applicationContext-spring-security-memory.xml** with a file editor and search for the DaoAuthenticationProvider bean. Add **id=authenticationProvider** to the bean.
  - b. Make sure that **SSL** is enabled on CAS.

8. Start the BA Server.

The BA Server is now configured to authenticate users against your central authentication server.

- [CAS Property Reference](#)

## CAS Property Reference

---

### CAS Properties

These properties mostly refer to CAS services:

| Property                 | Description  | Example   |
|--------------------------|--|---|
| cas.authn.provider       | <b>Required.</b> Security back-end that CAS should use. Valid values are memory, jdbc, or ldap   | ldap  |
| cas.login.url            | <b>Required.</b> CAS login URL.  | <code>\${cas.base.url}/login</code>                                 |
| cas.ticket.validator.url | <b>Required.</b> CAS ticket validator URL.   | <code>\${cas.base.url}</code>                                       |
| cas.logout.url           | <b>Required.</b> CAS logout URL. A <code>service.logout.url</code> will be appended to this URL. | <code>\${cas.base.url}/logout?url=</code>                           |
| cas.base.url             | URL under which all CAS services reside.   | <a href="https://localhost:8443/cas">https://localhost:8443/cas</a> |

### Pentaho Properties

These are service URLs that serve as callbacks from the CAS server into the BI Platform:

| Property                             | Description                                     | Example  |
|--------------------------------------|---|--|
| pentaho.service.url                  | <b>Required.</b> Processes CAS callback.        | <code>\${_pentaho.service.base.url}/j_spring_cas_security_check</code>       |
| pentaho.service.logout.url           | <b>Required.</b> URL to go to after CAS logout. | <code>\${_pentaho.service.base.url}/Home</code>                              |
| pentaho.service.solutions.system.dir | Path to pentaho-solutions/system.               | <code>/usr/local/pentaho/server/biserver-ee/pentaho-solutions/system/</code> |
| pentaho.service.lib.dir              | Path to webapp lib directory.                   | <code>/usr/local/tomcat/common/lib/</code>                                   |

|                                  |  |   |
|----------------------------------|--|---|
| pentaho.service.web.xml          | Path (including filename) of webapp's web.xml.                               | /usr/local/tomcat/conf/web.xml  |
| pentaho.service.appctx.cas.xml   | Path (including filename) of new applicationContext-spring-security-cas.xml. | /usr/local/pentaho/server/biserver-ee/pentaho-solutions/system/applicationContext-spring-security-cas.xml |
| pentaho.service.jsp.dir          | Path to directory containing webapp's JSPs.                                  | /usr/local/tomcat/webapps/pentaho/jsp/  |
| pentaho.service.spring.beans.xml | Path (including filename) of pentaho-spring-beans.xml.                       | /usr/local/pentaho/server/biserver-ee/pentaho-solutions/system/pentaho-spring-beans.xml                   |
| pentaho.service.base.url         | Service base URL.  | <a href="http://localhost:8080/pentaho">http://localhost:8080/pentaho</a>                                 |
| pentaho.service.pentaho.war.dir  | Webapp exploded WAR directory.   | /usr/local/tomcat/webapps/pentaho/  |
| pentaho.service.webinf.dir       | Path to webapp's WEB-INF directory.  | /usr/local/tomcat/webapps/pentaho/WEB-INF/  |

## Switch to Integrated Windows Authentication (IWA)

---

You must download this [patch JAR](#) before you switch to Integrated Windows Authentication.

This procedure requires Microsoft Windows Server 2008 R2, IIS 7.5, and Internet Explorer. If you are using different versions of any of this software, you may adjust the instructions to fit your needs.

Additionally, you will need to ensure that the following components of IIS are installed before continuing:

- Windows Authentication
- ISAPI Extensions
- ISAPI Filters
- JK 1.2 Connector (isapi\_redirect.dll)

Follow these instructions to switch to Integrated Windows Authentication in the BA Server.

1. Stop the BA Server, DI Server, and User Console processes.
2. Copy the downloaded patch JAR to the `/WEB-INF/lib/` directory inside of the deployed Pentaho WAR. For most deployments, this will be `/pentaho/server/biserver-ee/tomcat/webapps/pentaho/WEB-INF/lib/`
3. In your IIS configuration, disable anonymous authentication and enable Windows authentication for the site you are serving.
4. Edit the `/WEB-INF/web.xml` file inside of the deployed Pentaho WAR, and change the value of **fully-qualified-server-url** to the URL served by IIS, then save and close the file.
5. Edit the `/tomcat/conf/server.xml` file and set **tomcatAuthentication** to **false** in the **Connector** element for the connector with the **AJP** protocol.  
Note: If this is not already defined, then add it; the example below can be directly pasted into the file.

```
tomcatAuthentication="false"
```

6. Save and close the file, then edit `/pentaho-solutions/system/applicationContext-spring-security.xml`. Comment out this code block

```
<![CDATA[CONVERT_URL_TO_LOWERCASE_BEFORE_COMPARISON PATTERN_TYPE_APACHE_ANT
/**=securityContextHolderAwareRequestFilter,httpSessionContextIntegrationFilter,
httpSessionReuseDetectionFilter,logoutFilter,preAuthenticatedProcessingFilter,
authenticationProcessingFilter,basicProcessingFilter,requestParameterProcessingFilter,
anonymousProcessingFilter,pentahoSecurityStartupFilter,exceptionTranslationFilter,
filterInvocationInterceptor]]>
```

7. Copy and paste this code block immediately after the block you just commented out

```
<![CDATA[CONVERT_URL_TO_LOWERCASE_BEFORE_COMPARISON PATTERN_TYPE_APACHE_ANT
/**=httpSessionContextIntegrationFilter,httpSessionReuseDetectionFilter,
logoutFilter,preAuthenticatedProcessingFilter,authenticationProcessingFilter,
basicProcessingFilter,requestParameterProcessingFilter,anonymousProcessingFilter,
securityContextHolderAwareRequestFilter,pentahoSecurityStartupFilter,
exceptionTranslationFilter,filterInvocationInterceptor]]>
```

8. Find the **authenticationManager** providers list and add this line to the beginning of it:

```
<ref bean="preAuthAuthenticationProvider" />
```

9. Replace the **authenticationProcessingFilterEntryPoint** bean definition with the following:

```
<bean id="preAuthenticatedProcessingFilterEntryPoint"
      class="org.springframework.security.ui.preauth.
      PreAuthenticatedProcessingFilterEntryPoint" />
```

10. Find the **exceptionTranslationFilter** bean and replace its **authenticationEntryPoint** ref with:

```
<ref local="preAuthenticatedProcessingFilterEntryPoint" />
```

11. Ensure that you have configured Active Directory integration properly. Refer to your Active Directory documentation and [Manual MSAD Configuration](#) for more information.
12. Save and close the server.xml file.
13. Configure Internet Explorer such that your IIS server is in the **local intranet** security zone.
14. Start the BA Server.
15. Access the BA Server through Internet Explorer and ensure that it automatically logs in with the local user account.

Your system should now be configured to sign into the BA Server using local user account credentials.

#### Related information

<http://learn.iis.net/page.aspx/29/installing-iis-7-on-windows-server-2008-or-windows-server-2008-r2/>

<http://www.iisadmin.co.uk/?p=72>

<http://drumcoder.co.uk/blog/2010/may/05/iis-proxying-tomcat/>

<http://codesnip.net/iis7-integrated-windows-authentication-win-2008>

<http://tomcat.apache.org/tomcat-6.0-doc/config/ajp.html>

## Add Web Resource Authentication

---

To configure Web resource authentication in the BA Server to correspond with your user roles, follow the below instructions.

Note: These instructions are valid across all security DAOs.

1. Ensure that the BA Server is not currently running; if it is, run the **stop-pentaho** script.
2. Open a terminal or command prompt window and navigate to the `.../pentaho-solutions/system/` directory.
3. Edit the **applicationContext-spring-security.xml** file with a text editor.
4. Find and examine the following property: `<property name="objectDefinitionSource">`
5. Modify the regex patterns to include your roles. The **objectDefinitionSource** property associates URL patterns with roles. **RoleVoter** specifies that if any role on the right hand side of the equals sign is granted to the user, the user may view any page that matches that URL pattern. The default roles in this file are not required; you can replace, delete, or change them in any way that suits you.

You should now have coarse-grained permissions established for user roles.



## Restrict or Share Files and Folders

---

Access to files or folders can be refined using the Pentaho User Console. Each file or folder can either use the default permissions or you can tailor them for specific users and roles.

Prior to performing this task, you need to have determined whether you are going to use the default Pentaho roles, or created specific users and roles. You must also have successfully set up your security back end. Once you establish roles, you can share or restrict files and folders by role-type from the administration view within the User Console.

1. Log into the **User Console** using the administrator role.
2. From the **Browse Files** page, choose the folder you want to set permissions on from the **Folders** pane. If you want to set permissions on a specific file within that folder, click to highlight the file in the center **Files** pane.
3. Click **Properties** in the **Actions** pane on the right. The **Properties** window appears.
4. On the **Share** tab, highlight the **Role** that you want to set permissions for, then clear the check box next to **Inherits folder permissions**. The **Permissions for [Role]** field becomes accessible.
5. Select the permissions for that role using the check boxes and click **OK**.

The permissions are set for that file or folder and are associated with the selected role.

## Secure the User Console and BA Server

---

This section contains instructions and guidance for enhancing the security of the BA Server and User Console on an application server level via Secure Sockets Layer (SSL). SSL provides verification of server identity and encryption of data between clients and the BA Server.

- [Configure SSL \(HTTPS\) in the Pentaho User Console and BA Server](#)
- [Use the Apache Web Server \(httpd\) For Socket Handling](#)
- [How to Change the Administrator Role](#)
- [Secure SQL Access for Dashboards](#)

## Configure SSL (HTTPS) in the Pentaho User Console and BA Server

---

By default, the BA Server and Pentaho User Console are configured to communicate over HTTP. To switch to HTTPS, follow the instructions below that apply to your scenario.

- [Enable SSL in the BA Server With a Certificate Authority](#)
- [Enable SSL in the BA Server With a Self-Signed Certificate](#)
- [Change the BA Server Fully Qualified URL](#)

## Enable SSL in the BA Server With a Certificate Authority

---

If you already have an SSL certificate through a certificate authority such as Thawte or Verisign, all you have to do to use it with the Pentaho BA Server and User Console is configure your application server to use it. Apache provides documentation for configuring Tomcat for CA-signed certificates: <http://tomcat.apache.org/tomcat-6.0-doc/ssl-howto.html>. Just follow those procedures, and skip the sections below that deal with self-signed SSL certificates.

After the application server is configured to use your certificate, you must modify the base URL tokens for both the BA Server and the User Console. Make sure you follow the directions for [changing the BA Server Base URL](#); without executing those changes, your server will not work over HTTPS.

## Enable SSL in the BA Server With a Self-Signed Certificate

---

This process explains how to enable SSL in the BA Server with a self-signed certificate. These steps don't show how to generate a self-signed certificate, or how to configure Tomcat to use it. For more information on SSL certificates in Tomcat, consult [the Tomcat documentation](#), beginning with the *Quick Start* section.

- [Trust a Self-Signed Certificate](#)

## Trust a Self-Signed Certificate

---

The procedure below assumes that an SSL certificate is generated and Tomcat is configured to use it.

The instructions below explain how to complete the trust relationship between the BA Server (when it is configured for SSL) and the User Console.

1. Change to the home directory of the user account that starts the BA Server and Pentaho User Console processes or services.

```
cd ~
```

Using the default settings suggested by Pentaho, this will be `/home/pentaho/`.

2. Execute the following command, changing the storepass (**pass** in the example) and keypass (**pass2** in the example) accordingly:

```
keytool -export -alias tomcat -file tomcat.cer -storepass pass -keypass  
pass2 -keystore .keystore
```

3. Change to the `$PENTAHO_JAVA_HOME/jre/lib/security/` directory.

```
cd $PENTAHO_JAVA_HOME/jre/lib/security/
```

The `PENTAHO_JAVA_HOME` variable was established during your production installation procedure. If you are on Windows, environment variables are surrounded by percent signs, as in: `cd %PENTAHO_JAVA_HOME%\jre\lib\security\`. If you get an error about this path not being valid, then use `JAVA_HOME` instead of `PENTAHO_JAVA_HOME`.

4. Execute the following command, changing the alias (**tomcat** in the example), the file path to the certificate (the current user's home directory in the example), and the storepass (**pass** in the example) accordingly:

```
keytool -import -alias tomcat -file ~/tomcat.cer -keystore cacerts -  
storepass pass
```

Note: If the path to your certificate involves spaces, you must either escape the spaces (on Linux, Unix, and OS X), or put double quotes around the path (on Windows) in order for the command to work properly.

5. Execute the following command and make note of the MD5 sum for the **tomcat** entry:

```
keytool -list -keystore cacerts
```

6. Change back to the home directory of the user account that starts the BA Server and User Console, and run this command:

```
keytool -list -keystore .keystore
```

7. Compare the **tomcat** entry's MD5 sum to the one you generated previously and ensure that they match. If these sums do not match, you've made a mistake somewhere in the certificate trust process. Go through the steps again and ensure that you're working with the right user accounts and directories.

The BA Server is now configured to allow access via SSL.

## Change the BA Server Fully Qualified URL

---

If you switch from HTTP to HTTPS, you must also change the BA Server's tokenized fully qualified URL value to accommodate for the new port number. Follow the directions below to change the fully qualified URL.

1. Stop the BA Server if it is currently running.
2. Navigate to the `server/biserver-ee/tomcat/webapps/pentaho/WEB-INF/` directory.
3. Open the **web.xml** file with any text editor.
4. Locate this element and modify the port number to match your SSL-enabled port number.

```
<context-param>
  <param-name>fully-qualified-server-url</param-name>
  <param-value>http://localhost:8080/pentaho/</param-value>
</context-param>
```

5. Save and close the file.
6. Start the BA Server and make sure that it is available through HTTPS on the specified port.

The BA Server is now configured to communicate on an SSL-aware port.



## Use the Apache Web Server (httpd) For Socket Handling

---

Tomcat's socket handling abilities are not quite as robust as Apache httpd's are, especially when it comes to system error handling because Tomcat performs all its socket handling through the Java VM. Since Java is designed to be cross-platform, it lacks some system-specific optimizations; socket optimization is one such deficiency. In situations where the BA Server is hit with a large number of dropped connections, invalid packets, or invalid requests from invalid IP addresses, httpd would do a much better job of dropping these error conditions than Tomcat would. Therefore, you can improve BA Server security by fronting Tomcat with httpd.

A side-effect of this configuration is increased performance when delivering static content from the BA Server. For this reason, the same procedure below is covered in the section called [Optimize BA Server Performance](#). If you have already followed the Apache httpd procedure in that guide, there is no need to perform it again with the instructions below.

- [Use Apache httpd With SSL For Delivering Static Content](#)

## Use Apache httpd With SSL For Delivering Static Content

---

You can use the Apache httpd Web server to handle delivery of static content and facilitation of socket connections, neither of which is done efficiently through Tomcat alone, especially under heavy traffic or when accepting connections from the Internet.

1. Install Apache 2.2.x -- with SSL support -- through your operating system's preferred installation method. For most people, this will be through a package manager. It's also perfectly valid to download and install the reference implementation from <http://www.apache.org>. It is possible to use Apache 1.3, but you will have to modify the instructions on your own from this point onward.
2. If it has started as a consequence of installing, stop the Apache server or service.
3. Retrieve or create your SSL keys. If you do not know how to generate self-signed certificates, refer to the OpenSSL documentation. Most production environments have SSL certificates issued by a certificate authority such as Thawte or Verisign.
4. Check to see if you already have the Tomcat Connector installed on your system. You can generally accomplish this by searching your filesystem for **mod\_jk**, though you can also search your **httpd.conf** file for **mod\_jk**. If it is present, then you only need to be concerned with the Apache httpd configuration details and can skip this step. If it is not there, then the Tomcat Connector module needs to be installed. If you are using Linux or BSD, use your package manager or the Ports system to install **mod\_jk**. For all other platforms, visit the <http://www.apache.org/dist/tomcat/tomcat-connectors/jk/binaries/>, then click on the directory for your operating system. The module will be either an **.so** (for Linux, BSD, OS X, and Solaris) or **.dll** (for Windows) file. Save it to your Apache modules directory, which is generally **C:\Program Files\Apache Group\Apache2\modules\** on Windows, and **/usr/lib/apache2/modules/** on Unix-like operating systems, though this can vary depending on your Apache configuration.
5. Edit your **httpd.conf** file with a text editor and add the following text to the end of the file, modifying the paths and filenames as instructed in the comments:  
Note: Some operating systems use modular httpd configuration files and have unique methods of including each separate piece into one central file. Ensure that you are not accidentally interfering with an auto-generated **mod\_jk** configuration before you continue. In many cases, some of the configuration example below will have to be cut out (such as the **LoadModule** statement). In some cases (such as with Ubuntu Linux), **httpd.conf** may be completely empty, in which case you should still be able to add the below lines to it. Replace **example.com** with your hostname or domain name.

```
# Load mod_jk module
# Update this path to match your mod_jk location; Windows users should
change the .so to .dll
LoadModule      jk_module    /usr/lib/apache/modules/mod_jk.so
# Where to find workers.properties
# Update this path to match your conf directory location
JkWorkersFile  /etc/httpd/conf/workers.properties
# Should mod_jk send SSL information to Tomcat (default is On)
```

```

JkExtractSSL On
# What is the indicator for SSL (default is HTTPS)
JkHTTPSIndicator HTTPS
# What is the indicator for SSL session (default is SSL_SESSION_ID)
JkSESSIONIndicator SSL_SESSION_ID
# What is the indicator for client SSL cipher suit (default is SSL_CIPHER)
JkCIPHERIndicator SSL_CIPHER
# What is the indicator for the client SSL certificated (default is SSL_
CLIENT_CERT)
JkCERTSIndicator SSL_CLIENT_CERT
# Where to put jk shared memory
# Update this path to match your local state directory or logs directory
JkShmFile      /var/log/httpd/mod_jk.shm
# Where to put jk logs
# Update this path to match your logs directory location (put mod_jk.log
next to access_log)
JkLogFile      /var/log/httpd/mod_jk.log
# Set the jk log level [debug/error/info]
JkLogLevel     info
# Select the timestamp log format
JkLogStampFormat "[%a %b %d %H:%M:%S %Y] "
# Send everything for context /examples to worker named worker1 (ajp13)
# JkOptions indicates to send SSK KEY SIZE
JkOptions +ForwardKeySize +ForwardURISCompat -ForwardDirectories
# JkRequestLogFormat
JkRequestLogFormat "%w %V %T"
# Mount your applications
JkMount /pentaho/* tomcat_pentaho
# Add shared memory.
# This directive is present with 1.2.10 and
# later versions of mod_jk, and is needed for
# for load balancing to work properly
JkShmFile logs/jk.shm
<VirtualHost example.com
ServerName example.com
JkMount /pentaho default
JkMount /pentaho/* default
JkMount /sw-style default
JkMount /sw-style/* default
JkMount /pentaho-style default

```

```
JkMount /pentaho-style/* default
</VirtualHost>
```

6. In your Apache configuration, ensure that SSL is enabled by uncommenting or adding and modifying the following lines:

```
LoadModule ssl_module modules/mod_ssl.so
Include conf/extra/httpd-ssl.conf
```

7. Save and close the file, then edit `/conf/extra/httpd-ssl.conf` and properly define the locations for your SSL certificate and key:

```
SSLCertificateFile "conf/ssl/mycert.cert"
SSLCertificateKeyFile "conf/ssl/mycert.key"
```

8. Ensure that your SSL engine options contain these entries:

```
SSLOptions +StdEnvVars +ExportCertData
```

9. Add these lines to the end of the **VirtualHost** section:

```
JkMount /pentaho default
JkMount /pentaho/* default
JkMount /sw-style default
JkMount /sw-style/* default
JkMount /pentaho-style default
JkMount /pentaho-style/* default
```

10. Save and close the file, then create a **workers.properties** file in your Apache conf directory. If it already exists, merge it with the example configuration in the next step.
11. Copy the following text into the new **workers.properties** file, changing the location of Tomcat and Java, and the port numbers and IP addresses to match your configuration:  
Note: Remove the **workers.tomcat\_home** setting if you are using JBoss.

```
workers.tomcat_home=/home/pentaho/pentaho/server/biserver-ee/tomcat/
workers.java_home=/home/pentaho/pentaho/java/
worker.list=tomcat_pentaho
worker.tomcat_pentaho.type=ajp13
```

Apache httpd is now configured to securely and efficiently handle static content for Tomcat. You should now start Tomcat and httpd, then navigate to your domain name or hostname and verify that you can access the Pentaho Web application.

## How to Change the Administrator Role

---

The default administrator role in the BA Server is **Admin**. If you need to give this privilege level to a different role name, follow these instructions.

Note: Role names are case sensitive, so take special care when typing in the new role name.

1. Open the `/pentaho/server/biserver-ee/pentaho-solutions/system/pentaho.xml` file with a text editor.
2. Find the `<acl-voter>` element, and replace its `<admin-role>` property with the new administrator role (NewAdmin in the examples in this procedure).

```
<admin-role>NewAdmin</admin-role>
```

3. Find the `<acl-publisher>` element, and appropriately replace all instances of **Admin** in the properties inside of the `<default-acls>` and `<overrides>` elements.

```
<acl-entry role="NewAdmin" acl="ADMIN_ALL" />
```

4. Save the file, then open `applicationContext-spring-security.xml`
5. Find the `filterInvocationInterceptor` bean, and modify its `objectDefinitionSource` property accordingly. You may need to consult the Spring Security documentation to complete this step. The appropriate documentation is at <http://static.springsource.org/spring-security/site/reference.html>

```
<property name="objectDefinitionSource">
  <value>
    <![CDATA[
      CONVERT_URL_TO_LOWERCASE_BEFORE_COMPARISON
      ...
      \A/admin.*\Z=NewAdmin
      ...
    ]]>
  </value>
</property>
```

You have successfully changed the administrator role.

## Secure SQL Access for Dashboards

---

The Dashboard Designer has an SQL filter that allows greater control over a database query. By default, this feature is restricted to administrative users. To change these settings, follow the instructions below:

1. Ensure that the BA Server is not currently running; if it is, run the **stop-pentaho** script.
2. Open the `/pentaho-solutions/system/dashboards/settings.xml` file with a text editor.
3. Locate the following line and modify it accordingly:

```
<!-- roles with sql execute permissions -->  
<sql-execute-roles>Administrator</sql-execute-roles>
```

Note: Values are separated by commas, with no spaces between roles.

4. Locate the following line and modify it accordingly:

```
<!-- users with sql execute permissions -->  
<sql-execute-users>Administrator</sql-execute-users>
```

Note: Values are separated by commas, with no spaces between user names.

5. Save and close the text editor.
6. Restart the BA Server with the **start-pentaho** script.

The SQL filter function is now available in Dashboard Designer to the users and roles you specified.

## Remove Security by Allowing Anonymous Access

You can bypass the built-in security on the BA Server by giving all permissions to anonymous users. An "anonymousUser" is any user, either existing or newly created, that you specify as an all-permissions, no-login user, and to whom you grant the **Anonymous** role. The procedure below will grant full BA Server access to the **Anonymous** role and never require a login.

1. Stop the BA Server.

```
sudo /etc/init.d/pentaho stop
```

2. Open the `/pentaho/server/biserver-ee/pentaho-solutions/system/applicationContext-spring-security.xml` file with a text editor and ensure that a default anonymous role is defined. You may have changed this role, or it may not be properly defined for some other reason. Match your bean definition and property value to the example below.

```
<bean id="anonymousProcessingFilter" class="org.springframework.security.  
providers.anonymous.AnonymousProcessingFilter">  
  <!-- omitted -->  
  <property name="userAttribute" value="anonymousUser,Anonymous" />  
</bean>
```

3. This step allows Pentaho client tools to publish to the BA Server without having to supply a user name and password.

Find these two beans in the same file from the previous step.

- **filterSecurityInterceptor**
- **filterInvocationInterceptorForWS**

Locate the **objectDefinitionSource** properties inside the beans and match the contents to this code example.

Note: Make sure that there is a carriage return between COMPARISON and \A/ as shown below.

```
<bean id="filterInvocationInterceptor" class="org.springframework.security.  
intercept.web.FilterSecurityInterceptor">  
  <property name="authenticationManager">  
    <ref local="authenticationManager" />  
  </property>  
  <property name="accessDecisionManager">  
    <ref local="httpRequestAccessDecisionManager" />  
  </property>  
  <property name="objectDefinitionSource">
```

```

        <value>
            <![CDATA[ CONVERT_URL_TO_LOWERCASE_BEFORE_COMPARISON
\A/.*\Z=Anonymous,Authenticated ]]> </value>
        </property>
    </bean>

```

6. Save the file, then open `pentaho.xml` in the same directory.
7. Find the **anonymous-authentication** lines of the **pentaho-system** section, and define the anonymous user and role.

```

<pentaho-system>
<!-- omitted -->
    <anonymous-authentication>
        <anonymous-user>anonymousUser</anonymous-user>
        <anonymous-role>Anonymous</anonymous-role>
    </anonymous-authentication> <!-- omitted -->
</pentaho-system>

```

8. In the same directory, open the **repository.spring.properties** file.
  - a. Find the **singleTenantAdminAuthorityName** and replace the value with **Anonymous**.
  - b. Find the **singleTenantAdminUserName** and replace the value with the name **<your anonymous user>**.
  - c. Save the file and close the text editor.
9. In the same directory, open the **pentahoObjects.spring.xml** file.
  - a. Find all references to the **bean id="Mondrian-UserRoleMapper"** and make sure that the only one that is uncommented (active) is this one:

```

<bean id="Mondrian-UserRoleMapper"
    name="Mondrian-SampleUserSession-UserRoleMapper"
    class="org.pentaho.platform.plugin.action.mondrian.mapper.
        MondrianUserSessionUserRoleListMapper"
    scope="singleton">
    <property name="sessionProperty" value="MondrianUserRoles" />
</bean>

```

- b. If you have made any changes to **pentahoObjects.spring.xml**, save and close the file.

You have now effectively worked around the security features of the BA Server.

If you are using the relational metadata database model, refer to [Remove Security from Metadata Domain Repository](#) for the next few steps.

- [Remove Security from Data Source Management](#)



## Remove Security from Data Source Management

---

This procedure changes your data source management so that an anonymous user can access it. These steps are necessary to completely remove security from the BA Server; however, this procedure does not remove **all** security. Start with [Remove Security by Allowing Anonymous Access](#) if you need to remove all security.

1. If you need to, stop the BA Server.
2. Open `/pentaho/server/biserver-ee/pentaho-solutions/system/data-access/settings.xml` file with a text editor.
  - a. Find this line `<data-access-roles>Administrator</data-access-roles>` in the file and change this text:

**Administrator** to **Anonymous**

- a. Find this line `<data-access-view-roles>Authenticated,Administrator</data-access-view-roles>` in the file and change this text:

**Authenticated,Administrator** to **Anonymous**

- b. Find this line `<data-access-view-users>suzy</data-access-view-users>` and change this text:

**suzy** to **anonymousUser**

- c. Find this line `<data-access-datasource-solution-storage>admin</data-access-datasource-solution-storage>` and change this text:

**admin** to **anonymousUser**

3. Save and close the file.
4. Restart the BA Server.

## Troubleshooting

---

Adjusting authorization and authentication settings will often involve making multiple configuration changes without the benefit of testing each of them individually. Your first attempt at implementing different security access or performing intensive user and role modifications will probably not work perfectly. Below are some tips for adjusting log file output, and examining logs for signs of configuration errors.

- [Increase Security Log Levels in the BA Server](#)
- [Log Output Analysis](#)
- [LDAP Roles Are Not "Admin" and "Authenticated"](#)
- [With LDAP Authentication, the PDI Repository Explorer is Empty](#)
- [LDAP Incorrectly Authenticates User IDs That Do Not Match Letter Case](#)

## Increase Security Log Levels in the BA Server

---

The security logging facilities of the BA Server are set to ERROR by default, which is not always verbose enough for troubleshooting and testing. The below procedure explains how to increase the level of detail in the BA Server logs that deal with security-related messages.

1. Stop the BA Server.

```
sh /usr/local/pentaho/server/biserver-ee/stop-pentaho.sh
```

2. Open the `/pentaho/server/biserver-ee/tomcat/webapps/pentaho/WEB-INF/classes/log4j.xml` file with a text editor.
3. Use XML comments (`<!-- -->`) to disable all of the **Threshold** parameters in all of the **appender** elements.
4. Change the priority value in the `<root>` section to one of: **WARN**, **ERROR**, **FATAL**, or **DEBUG**, depending on which level you prefer.

```
<root>
  <priority value="DEBUG" />
  <appender-ref ref="PENTAHOCONSOLE"/>
  <appender-ref ref="PENTAHOFILE"/>
</root>
```

5. Add the following loggers directly above the root element:

```
<!-- all Spring Security classes will be set to DEBUG -->
<category name="org.springframework.security">
  <priority value="DEBUG" />
</category>

<!-- all Pentaho security-related classes will be set to DEBUG -->
<category name="org.pentaho.platform.engine.security">
  <priority value="DEBUG" />
</category>
<category name="org.pentaho.platform.plugin.services.security">
  <priority value="DEBUG" />
</category>
```

6. Save and close the file, then edit the Spring Security configuration file that corresponds with your security back end in the `/pentaho/server/biserver-ee/pentaho-solutions/system/` directory.

The file will be one of:

- applicationContext-spring-security-memory.xml
- applicationContext-spring-security-jdbc.xml
- applicationContext-spring-security-ldap.xml

10. Find the **daoAuthenticationProvider** bean definition, and add the following property anywhere inside of it (before the `</bean>` tag):

```
<property name="hideUserNotFoundExceptions" value="false" />
```

11. Save the file and close the text editor.
12. Start the BA Server.

```
sh /usr/local/pentaho/server/biserver-ee/start-pentaho.sh
```

BA Server security logging is now globally set to DEBUG, which is sufficiently verbose for debugging security configuration problems. All BA Server messages will be collected in the `/pentaho/server/biserver-ee/logs/pentaho.log` file.

When you are finished configuring and testing the BA Server, you should adjust the log levels down to a less detailed level, such as ERROR, to prevent `pentaho.log` from growing too large.

- [Enable Extra LDAP Security Logging](#)

## Enable Extra LDAP Security Logging

---

If you need yet more LDAP-related security details in **pentaho.log**, or if you are specifically having difficulty with LDAP authentication configuration, follow the below process.

Note: These instructions are for testing and pre-production only. User names and passwords will be displayed in the log file in plain text.

1. Stop the BA Server.

```
sh /usr/local/pentaho/server/biserver-ee/stop-pentaho.sh
```

2. Open the `/pentaho/server/biserver-ee/pentaho-solutions/system/applicationContext-spring-security-ldap.xml` file with a text editor.
3. Change the reference in the first **constructor-arg** property of the **daoAuthenticationProvider** element to **LdapAuthenticatorProxy**

```
<constructor-arg>
    <ref bean="ldapAuthenticatorProxy" />
</constructor-arg>
```

4. Save the file, then create a new file called `applicationContext-logging.xml` in the same directory.
5. Copy the following text into the new file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN" "http://www.
springframework.org/dtd/spring-beans.dtd">
<beans>
    <bean id="ldapAuthenticatorProxy" class="org.springframework.aop.
framework.ProxyFactoryBean">
        <property name="proxyInterfaces">
            <value>org.springframework.security.providers.ldap.
LdapAuthenticator</value>
        </property>
        <property name="target">
            <ref bean="authenticator" />
        </property>
        <property name="interceptorNames">
            <list>
                <value>loggingAdvisor</value>
            </list>
        </property>
    </bean>
</beans>
```

```

        </property>
    </bean>
    <bean id="loggingAdvisor" class="org.springframework.aop.support.
RegexMethodPointcutAdvisor">
        <property name="advice">
            <ref local="loggingInterceptor" />
        </property>
        <property name="pattern">
            <value>.*</value>
        </property>
    </bean>
    <bean id="loggingInterceptor" class="org.pentaho.platform.engine.
security.LoggingInterceptor" />
</beans>

```

6. Save the file, then open `pentaho-spring-beans.xml`.

7. Add the following import line to the list of files:

```
<import resource="applicationContext-logging.xml" />
```

8. Save and close the file, then start the BA Server.

```
sh /usr/local/pentaho/server/biserver-ee/start-pentaho.sh
```

You will now have verbose LDAP-specific log messages in **pentaho.log** that include login credentials for every user that tries to log in.

## Log Output Analysis

---

The information you need to look for in **pentaho.log** in order to troubleshoot security configuration problems should be fairly self-evident. If you are having trouble, consult the examples below.

When you request a page that is protected but you are not yet logged in, you should see an exception in the log which looks like this:

```
DEBUG [ExceptionTranslationFilter] Access is denied (user is anonymous);
    redirecting to authentication entry point org.springframework.
security.AccessDeniedException:
    Access is denied
```

When the user name and/or password doesn't match what's stored in the back end, you should see a log message like this:

```
WARN [LoggerListener] Authentication event
    AuthenticationFailureBadCredentialsEvent: suzy; details:
    org.springframework.security.ui.WebAuthenticationDetails@fffd148a:
RemoteIpAddress: 127.0.0.1;
    SessionId: 976C95033136070E0200D6DA26CB0277; exception: Bad
credentials
```

When the user name and password match, you should see a log message that looks like the example below. After the **InteractiveAuthenticationSuccessEvent**, one of the filters will show the roles fetched for the authenticated user. Compare these roles to the page-role mapping found in the **filterInvocationInterceptor** bean in **applicationContext-spring-security.xml**.

```
WARN [LoggerListener] Authentication event InteractiveAuthenticationSuccessEvent:
    suzy; details: org.springframework.security.ui.
WebAuthenticationDetails@fffd148a: RemoteIpAddress:
    127.0.0.1; SessionId: 976C95033136070E0200D6DA26CB0277 DEBUG
    [HttpSessionContextIntegrationFilter] SecurityContext stored to
HttpSession:
    'org.springframework.security.context.SecurityContextImpl@2b86afeb:
Authentication:
    org.springframework.security.providers.
UsernamePasswordAuthenticationToken@2b86afeb: Username:
```

```
org.springframework.security.userdetails ldap.
LdapUserDetailsImpl@d7f51e; Password: [PROTECTED];
Authenticated: true; Details: org.springframework.security.ui.
WebAuthenticationDetails@fffd148a:
RemoteIpAddress: 127.0.0.1; SessionId:
976C95033136070E0200D6DA26CB0277; Granted
Authorities: ROLE_CTO, ROLE_IS, ROLE_AUTHENTICATED'
```

If you are troubleshooting LDAP problems, look for log output similar to this:

```
DEBUG [DirMgrBindAuthenticator] (LoggingInterceptor) Return value: LdapUserInfo:
org.springframework.security.providers.ldap.
LdapUserInfo@1f31c64[dn=uid=suzy,ou=users,ou=system,attributes={mail=mail:
suzy.pentaho@pentaho.org, uid=uid: suzy, userpassword=userpassword:
[B@e17c9c,
businesscategory=businesscategory: cn=cto,ou=roles,ou=system,
cn=is,ou=roles,ou=system,
objectclass=objectClass: organizationalPerson, person,
groupOfUniqueNames,
inetOrgPerson, top, uniquemember=uniquemember: cn=cto, ou=roles, cn =
is , ou = roles,
sn=sn: Pentaho, cn=cn: suzy}]
```



## LDAP Roles Are Not "Admin" and "Authenticated"

---

You must not use **Admin** and **Authenticated** roles in your LDAP. Instead you must configure your system to use **pentahoAdmins** and **pentahoUsers** or other easily identifiable role names. Edit `/pentaho-solutions/system/applicationContext-spring-security.xml`. At the bottom of this file, you will find a number of lines that look like: `A/docs/. *Z=Anonymous,Authenticated`.

These are entries for URL Security. They are regular expressions to match a path on the browser's URL that require the user to be a member of the defined role to gain access. In the example above, both **Anonymous** and **Authenticated** get access. In the example above, use **pentahoUsers** in the place of **Authenticated**. by entering `A/docs/. *Z=Anonymous,pentahoUsers`. For all entries that show **Authenticated**, replace it with **pentahoUsers** or your chosen name. Replace **Admin** with **pentahoAdmins** or your chosen name. For the change from **Authenticated** to **pentahoUsers** replace all occurrences. For **Admin** to **pentahoAdmins** you need to be a little more careful because there are some entries that look like this: `A/admin.*Z=pentahoAdmins`.

Edit the `/pentaho-solutions/system/repository.spring.xml` file and change:

```
<bean id="singleTenantAuthenticatedAuthorityName" class="java.lang.String">
  <constructor-arg value="Authenticated" />
</bean>
```

to:

```
<bean id="singleTenantAuthenticatedAuthorityName" class="java.lang.String">
  <constructor-arg value="pentahoUsers" />
</bean>
```

and:

```
<bean id="singleTenantAdminAuthorityName" class="java.lang.String">
  <constructor-arg value="Admin" />
</bean>
```

to:

```
<bean id="singleTenantAdminAuthorityName" class="java.lang.String">  
    <constructor-arg value="pentahoAdmins" />  
</bean>
```

## With LDAP Authentication, the PDI Repository Explorer is Empty

---

If you log into a solution repository from Spoon before you switch the authentication to LDAP, then the repository IDs and security structures will be broken. You won't see an error message, but the solution repository explorer will be empty and you won't be able to create new folders or save PDI content to it. To fix the problem, you will have to delete the security settings established with the previously used authentication method, which will force the DI Server to regenerate them for LDAP.

**CAUTION:**

Following this procedure will destroy any previously defined DI repository users, roles, and access controls. You should back up the files that you delete in these instructions.

1. Stop the DI Server
2. Delete the security and default directories from the following directory: `/pentaho-solutions/system/jackrabbit/repository/workspaces/`
3. Start the DI Server

You should now have a proper LDAP-based DI Repository that can store content and create new directories.

## LDAP Incorrectly Authenticates User IDs That Do Not Match Letter Case

---

Some LDAP implementations are case-insensitive, most notably Microsoft Active Directory. When using one of these LDAP distributions as a BA Server authentication back end, you might run into an issue where a valid user name with invalid letter cases will improperly validate. For instance, if **Bill** is the valid user ID, and someone types in **bill** at the User Console login screen, that name will authenticate, but it might have improper access to parts of the BA Server.

The fix for this is documented: [LDAP Authenticates User IDs That Do Not Match Case](#).

- [LDAP Authenticates User IDs That Do Not Match Case](#)

## LDAP Authenticates User IDs That Do Not Match Case

---

Some LDAP implementations are case-insensitive for user names, most notably Microsoft Active Directory. You might run into an issue where a user name typed into the login screen does not exactly match the letter case of that user's ID in the directory, but the server will authenticate it anyway and may give the user improper access to parts of the BA Server. For example, if **Bill** is the valid user ID, and someone types in **bill** at the User Console login screen, the incorrectly typed one will authenticate, but it may have improper access to parts of the BA Server.

Follow these instructions to force case-sensitivity and fix this potential security risk.

1. Stop the BA Server.
2. Edit the `/pentaho/server/biserver-ee/pentaho-solutions/system/applicationContext-spring-security-ldap.xml` file.
3. Find the **daoAuthenticationProvider** bean, and below the last `</constructor-arg>` therein, and add the **<property>** definition shown in the example:

```
<property name="userDetailsContextMapper">
    <ref local="ldapContextMapper" />
</property>
```

4. After the `</bean>` tag for **daoAuthenticationProvider**, add the following bean definition, changing the **ldapUsernameAttribute** from **samAccountName** to the value that matches your environment:

```
<bean id="ldapContextMapper" class="org.pentaho.platform.engine.security.
UseridAttributeLdapContextMapper">
    <property name="ldapUsernameAttribute" value="samAccountName" />
</bean>
```

5. Start the BA Server.

The BA Server will now force case sensitivity in LDAP user names.