



The Pentaho Security Guide



This document is copyright © 2011 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

If you have questions that are not covered in this guide, or if you would like to report errors in the documentation, 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.

For information about instructor-led training on the topics covered in this guide, visit <http://www.pentaho.com/training>.

Limits of Liability and Disclaimer of Warranty

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

Pentaho (TM) and the Pentaho logo are registered trademarks of Pentaho Corporation. All other trademarks are the property of their respective owners. Trademarked names may appear throughout this document. Rather than list the names and entities that own the trademarks or insert a trademark symbol with each mention of the trademarked name, Pentaho 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.

Company Information

Pentaho Corporation
Citadel International, Suite 340
5950 Hazeltine National Drive
Orlando, FL 32822
Phone: +1 407 812-OPEN (6736)
Fax: +1 407 517-4575
<http://www.pentaho.com>

E-mail: communityconnection@pentaho.com

Sales Inquiries: sales@pentaho.com

Documentation Suggestions: documentation@pentaho.com

Sign-up for our newsletter: <http://community.pentaho.com/newsletter/>

Contents

Introduction: Configuring Security.....	5
Security Overview.....	6
Supported Technologies.....	7
Security Implementation Checklist.....	8
BA Server User Authentication.....	9
Pentaho (Default).....	9
Switching to LDAP.....	9
Microsoft Active Directory Configuration.....	10
LDAP Configuration in the Pentaho Enterprise Console.....	11
LDAP Properties.....	12
Forcing Case-Sensitivity in LDAP.....	13
Switching to JDBC.....	14
Switching to an LDAP/JDBC Hybrid.....	14
Implementing Single Sign-On.....	16
Switching to Central Authentication Service (CAS).....	16
Switching to Integrated Windows Authentication (IWA).....	17
Assigning Permissions in the Pentaho User Console.....	18
Permissions Settings.....	19
BA Server Content Authorization.....	20
User and Role Configuration.....	20
Adding Users.....	20
Editing User Information.....	20
Deleting Users.....	20
Adding Roles.....	21
Editing Roles.....	21
Deleting Roles.....	21
Assigning Users to Roles.....	21
How to Change the Administrator Role.....	22
Implementing Nested Roles in LDAP.....	22
Resetting or Creating a new Pentaho Enterprise Console User.....	23
Adding Web Resource Authentication.....	24
Domain Object Authorization.....	24
Reapplying the Default Access Control Lists.....	25
Configuring SQL Filters for Dashboards.....	26
Assigning Data Source Permissions for the Pentaho User Console.....	27
Securing the Pentaho Enterprise Console and BA Server.....	28
Configuring SSL (HTTPS) in the Pentaho Enterprise Console and BA Server.....	28
Enabling SSL in the BA Server With a Certificate Authority.....	28
Enabling SSL in the BA Server With a Self-Signed Certificate.....	28
Changing the BA Server Base URL.....	29
Enabling SSL in the Pentaho Enterprise Console.....	29
Changing Default Enterprise Console Security Settings.....	30
Changing the Admin Credentials for the Pentaho Enterprise Console.....	32
Creating a Custom Login Module.....	32
Using the Apache Web Server (httpd) For Socket Handling.....	33
httpd Configuration With Tomcat.....	33
Removing Security.....	36
Switching the Metadata Domain Repository.....	37
Switching to a File-Based Solution Repository.....	37
Metadata Security.....	39
Configuring the Security Service.....	39
Adding Column-Level Security Constraints.....	39
Adding Global Row-Level Security Constraints.....	40
MQL Formula Syntax For Global Constraints.....	40

Adding User or Role Row-Level Security Constraints.....	41
MQL Formula Syntax For User and Role Row-Level Constraints.....	42
Restricting Metadata Models to Specific Client Tools.....	42
Analysis Schema Security.....	44
Restricting Access to Specific Members.....	44
Mondrian Role Mapping in the BA Server.....	45
The Mondrian One-To-One UserRoleMapper.....	45
The Mondrian SampleLookupMap UserRoleMapper.....	45
The Mondrian SampleUserSession UserRoleMapper.....	45
Restricting ROLAP Schemas Per User.....	46
Using Security Information In Action Sequences.....	47
Troubleshooting.....	48
Miscellaneous Troubleshooting Tips.....	48
Increasing Security Log Levels in the BI Platform.....	48
Enabling Extra LDAP Security Logging.....	49
Log Output Analysis.....	50
LDAP Roles Are Not "Admin" and "Authenticated".....	51
With LDAP Authentication, the PDI Repository Explorer is Empty.....	51
LDAP incorrectly authenticates user IDs that don't match letter case.....	52

Introduction: Configuring Security

This guide helps system administrators configure Pentaho BI Platform security, including both its native security features and supported third-party security frameworks.

The Pentaho BI Platform supports most popular user authentication and access/authorization technologies. All Pentaho administrators will have to establish users and roles that match their organizational hierarchy, but few will need to change the method of user authentication.

To fully grasp the concepts and tasks involved in configuring security, you should be aware of a few technical terms:

- **Authentication:** The process of confirming that the user requesting access is the user that they claim to be. This is often done by presenting a user identifier (a username) paired with a secret known only to that user (a password), but can sometimes involve certificates or other means of establishing identity. In this documentation, authentication is synonymous with **login**.
- **Authorization:** The process of deciding if the authenticated user is allowed to access the information or functionality he is requesting. A software system can protect itself at multiple levels. In the Pentaho BI Platform, pages in the Web-based user interface can be protected. In addition, objects within the Pentaho solution repository, such as folders and action sequences, can be protected using access control lists (ACLs).
- **Security backend:** A repository of usernames, passwords, and roles. The repository can be a flat file, an RDBMS accessed via JDBC, or a directory server accessed via LDAP.
- **Security data access object (DAO):** A method of accessing the security backend. Examples of a security data access object are JDBC, Pentaho (Hibernate-based), and LDAP. (Both JDBC and Pentaho security data access objects talk to an RDBMS security backend, although they go about it in slightly different ways.)

Refer only to the sections below that apply to your situation.

Security Overview

The first half of this guide covers low-level BA Server and Pentaho Enterprise Console configuration.

- **Switch to a different authentication backend:** Many Pentaho admins prefer to use their own authentication method such as LDAP, Active Directory, single sign-on, or custom security tables in an existing RDBMS. There is also information about how to remove all authentication support in rare situations where logins are disfavored.
- **Access to the Pentaho Enterprise Console:** Because Enterprise Console has many options for low-level BA Server configuration, you may want to restrict access to it. You might also want to enable https support to ensure secure communication.

The second half of this guide covers the many ways to restrict access to Pentaho content. These restrictions only apply to the Pentaho User Console and its client tools: Analyzer, Dashboard Designer, and Interactive Reporting, as well as any extension, plugin, or service that uses the BA Server or BI Platform for user authentication (such as CDF, or an action sequence).



Note: There is no way to restrict access to any content that is not published to the BA Server. So if you create a report in Report Designer but do not publish it to the BA Server, there are no access restrictions available for it. Also, anyone with local user access to the BA Server solution repository (the pentaho-solutions directory) or the database that stores solutions and schedules (hibernate and quartz) will be able to see all published BA Server content. The restrictions described below only apply to users accessing the Pentaho User Console through a Web browser (over a private intranet or the public Internet).

- **Access to "raw" data sources:** Using the Pentaho Enterprise Console, you can restrict access to Pentaho data sources that have been established through Enterprise Console. However, any user can use a flat file (CSV, XML, XLS) to establish a data source, and any user that knows JDBC or JNDI connection information to a database can establish a data source on his own at the application server (global) or client tool (local) level. This is a matter of internal security at your organization and is beyond the scope of Pentaho software.
- **Access to metadata models:** Pentaho metadata models have fine-grained options for restricting user access. Metadata models are most commonly used as data sources in Interactive Reporting and Dashboard Designer, though Report Designer and PDI can access them as well. You can add access restrictions with Pentaho Metadata Editor; afterward, you must publish the model to the BA Server in order for your changes to take effect.
- **Access to ROLAP (Mondrian) schemas:** ROLAP schemas created with Schema Workbench and published to the BA Server can restrict access to part or all of a schema based on user roles. These are Mondrian roles, however, and are by default not connected in any way to the BA Server. In order to use BA Server roles in a Mondrian schema, you must configure one of three methods of role mapping. ROLAP schemas are used as data sources in Analyzer and Dashboard Designer, but can also be accessed by Report Designer and PDI.

Supported Technologies

The Pentaho BI Platform uses the Spring Security infrastructure, which can work with several common security backends:

- Flat file
- Active Directory, LDAP, or other directory server
- RDBMS (security tables in an existing database)

Pentaho's default security backend is an RDBMS, and its Hibernate-based security data access object is referred to as **Pentaho**. The security tables and access control lists are installed by default with the BI Platform, and can be easily configured through the Pentaho Enterprise Console's graphical user interface. This is a tested, reasonably secure method of managing resource authorization and user authentication, so there should be no reason to change to another security backend unless you've already deployed one.



Note: Switching to a non-default security backend means that you will have to hand-edit some BI Platform configuration files in order to change the security data access object. It also means that you will be unable to use the Pentaho Enterprise Console to manage users and roles.

Security Implementation Checklist

Step	Procedure	Done
Step 1	Develop a solid plan for your security system. For example, you must have the appropriate security backend (a directory server, for instance) in place and operational.	
Step 2	Determine user roles. What roles (out of potentially many) will have meaning in the Pentaho BI Platform? For example, you might have roles (or groups) that are used by other applications in your company. You could reuse those roles or define new ones for use in the Pentaho BI Platform. If you already have a BI_USER role, you could tell Pentaho to use that existing role.	
Step 3	Determine which roles should have access to particular URLs. These roles will be used to define Web resource authorization. For example, what role will be considered the Pentaho administrator? Pentaho has reasonable default Web resource authorization settings, so you probably won't need to change the URLs that are protected. You will have to change the roles that are allowed to access each URL, however.	
Step 4	Determine which roles should have which permissions to particular action sequences in the solution repository. These roles will be used to define domain object authorization. For example, will role A be allowed to execute action sequences in folder X?	
Step 5	Configure the security DAO. Leave it set to the default Pentaho security DAO or switch to JDBC, LDAP, or hybrid LDAP+JDBC.	
Step 6	If you'd like to use a role prefix, define one. By default, there is no role prefix.	
Step 7	Define the Pentaho administrator role.	
Step 8	Define the domain object authorization rules. These refer to the roles defined in step 5 above.	
Step 9	Apply the access control lists (ACLs). This step is a batch operation and will remove any custom permissions created via the Admin Permissions section, or via the Pentaho User Console.	
Step 10	Define the Web resource authorization rules in the filterInvocationInterceptor bean in /pentaho/server/biserver-ee/pentaho-solutions/system/applicationContext-spring-security.xml file. These refer to the roles defined in step 3 above.	
Step 11	Set plugin security. There are various plugins that have authorization settings.	
Step 12	Restrict data source editing/creation by editing /pentaho-solutions/system/data-access-plugin/settings.xml.	
Step 13	Configure Metadata security using Pentaho Metadata Editor.	
Step 14	Configure Mondrian security by configuring the Mondrian-UserRoleMapper bean ID in pentahoObjects.spring.xml.	
Step 15	Setup a secure connection between the Pentaho Enterprise Console and the BA Server.	
Step 16	If users will be publishing content to the BA Server, set the publish password in /pentaho-solutions/system/publisher_config.xml.	
Step 17	Setup a trust between the Pentaho Enterprise Console and the BA Server. Users must supply this password in addition to their usual user name and password.	
Step 18	If you are changing the Admin role on a DI Server, edit repository.spring.xml.	

BA Server User Authentication

By default, the BI Platform establishes roles, users, and initializes a basic configuration for the built-in Pentaho security data access object. You will almost certainly want to customize the roles and delete or modify the default users and add your own; at most, you will want to use your own LDAP, JDBC, or CAS (single sign-on) authentication mechanism with the BI Platform. This section explains these tasks in detail.



Note: Before you proceed with any instructions in this section, you should ensure that your BI Platform instance is working on a basic level. This initial verification will make it easier for you to retrace your steps later if you end up with a non-working configuration. You should also back up Pentaho Business Analytics or BA Server directory (if you installed via the graphical installation utility, or unpacked pre-configured archive packages), or your Pentaho WAR, and your pentaho-solutions, and enterprise-console directories (if you built the Pentaho WAR and performed a manual deployment to an existing application server).

Pentaho (Default)

The Pentaho security data access object is a custom Hibernate-based user/password DAO that reads and writes usernames, passwords, and roles to a relational database via Hibernate object-relational mapping.

You do not have to do anything to initialize the Pentaho data access object; it is enabled by default. However, you will almost certainly need to establish roles and users to match your organizational structure. Instructions for creating and modifying roles and users are in the Authorization subsection below.



Note: While Pentaho's default security system is good for evaluation and small production deployments (less than 100 users), Pentaho recommends a dedicated authentication provider such as LDAP or MSAD for large production environments.

Switching to LDAP

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

Follow the below instructions to switch the BA Server's authentication backend from the Pentaho data access object to LDAP.



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 Pentaho Enterprise Console.
2. Open the `/pentaho-solutions/system/pentaho-spring-beans.xml` file with a text editor, find the following two adjacent lines, and change the **hibernate** segment to **ldap** in each:

```
<import resource="applicationContext-spring-security-hibernate.xml" />
<import resource="applicationContext-pentaho-security-hibernate.xml" />
```

3. Save and close the file, then edit the `/pentaho-solutions/system/data-access/settings.xml` file and modify the user and role settings to match your LDAP configuration:

```
<!-- roles with data access permissions -->
<data-access-roles>pentahoAdmins</data-access-roles>
<!-- users with data access permissions -->
<!--
<data-access-users></data-access-users>
-->
<!-- roles with datasource view permissions -->
<data-access-view-roles>pentahoUsers,pentahoAdmins</data-access-view-roles>
<!-- users with datasource view permissions -->
<!-- <data-access-view-users></data-access-view-users> -->
<!-- default view acs for user or role -->
<data-access-default-view-acls>31</data-access-default-view-acls>
```

4. 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 **Admin** and **Authenticated** role values to match the appropriate roles in your LDAP configuration:
 - pentaho.xml

- repository.spring.xml
 - applicationContext-spring-security.xml
5. Using the command line or your preferred database administration tool, drop the **hibernate.pro_files** and **hibernate.pro_acls_list** columns from the **hibernate** table in your Pentaho solution repository database.
 6. Delete the /tomcat/work/ and /tomcat/temp/ directories.
 7. Start the BA Server and Pentaho Enterprise Console.
 8. Log into the Pentaho Enterprise Console.
 9. Configure the Pentaho LDAP connection as explained in [LDAP Properties](#) on page 12.
 10. Go to **Configuration > Web Settings**.
 11. Under **Authentication** select **LDAP Based** from the drop-down list and click **Submit**.
The BA Server should restart automatically.

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

Microsoft Active Directory Configuration

The Pentaho BI Platform 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.



Note: The information in this section also applies to configuring Active Directory for Pentaho Data Integration.

Binding

MSAD allows you to uniquely specify users in two ways, in addition to the standard DN. If you're not having luck with the standard DN, 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 penthoadmin@mycompany.com:

File: **applicationContext-security-ldap.properties**

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

Windows domain notation example for MYCOMPANY\penthoadmin:

File: **applicationContext-security-ldap.properties**

```
contextSource.providerUrl=ldap://mycompany\389
contextSource.userDn=MYCOMPANY\penthoadmin
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 username. 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> <constructor-arg index="2">
    <ref local="contextSource" />
  </constructor-arg>
  <property name="searchSubtree" value="true"/>
</bean>
```

LDAP Configuration in the Pentaho Enterprise Console



Note: The LDAP section of Enterprise Console is only for the BA Server; the Data Integration Server is not affected by LDAP settings defined here.

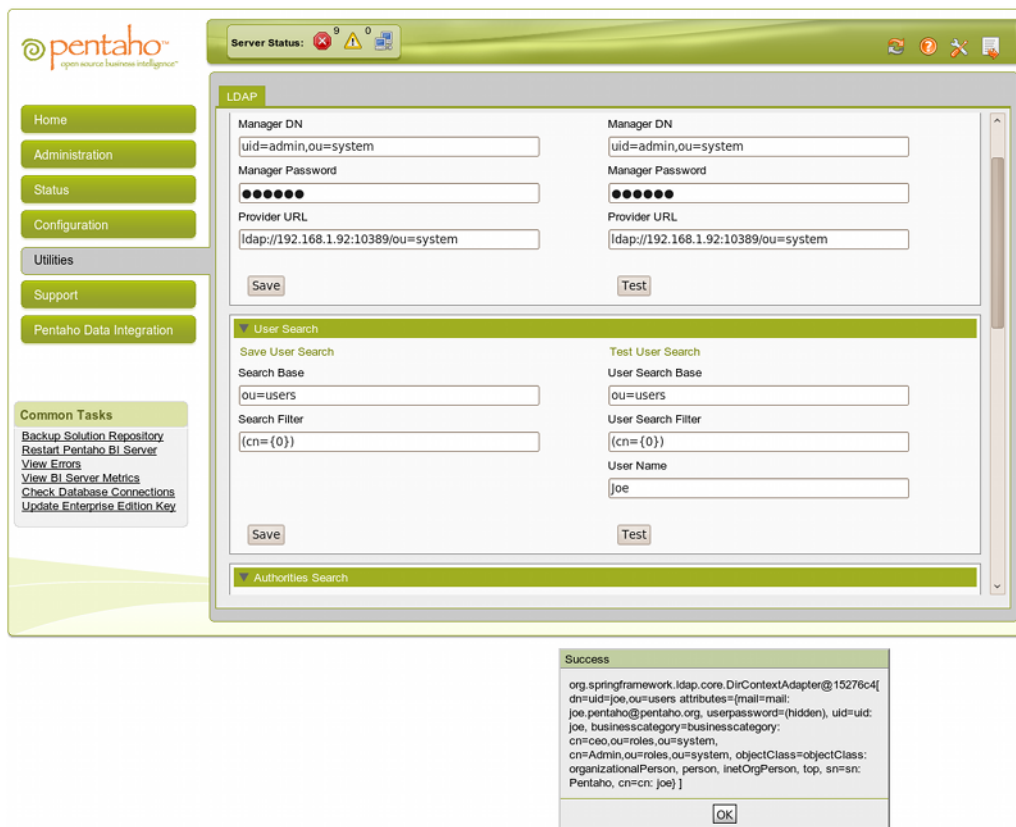
The Pentaho Enterprise Console has an LDAP section in the **Utilities** category. The fields in this screen should be relatively intuitive, but are explained below in more detail in case there are options that you are not familiar with.

You can view property values and operate on them; specific operations are exposed that allow you execute a "dry run" of a proposed security configuration. For each of the four property groups, there are two operations provided: test and save. Test allows you to test the property values in a property group without saving them. Save writes the property values in a property group to permanent storage.

Test and Save Operations

Testing does not rely on the BA Server; however, even though test operations do not test "live" objects, the test environment mimics the runtime environment as closely as possible. While save operations write property values to permanent storage, those property values are not visible to a running BA Server, so you must restart it after each save.

A successful test operation should produce a small dialogue with the query results in it:



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 Pentaho Enterprise Console for the BA Server (the LDAP options in PEC 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=</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 usernames that are entered in the Pentaho login dialog box.

 **Note:** The `{0}` token will be replaced by the username 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 username) for user searches	<code>userSearch.searchBase=CN=Users</code>
<code>userSearch.searchFilter</code>	Filter (by username) for user searches. The attribute you specify here must contain the value that you want your users to log into Pentaho with. Active Directory usernames are represented	<code>userSearch.searchFilter=(sAMAccountName=</code>

LDAP Property	Purpose	Example
	by sAMAccountName ; full names are represented by displayName .	

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 username 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 username) 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:1.2.8
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 false , the search must include the current object only.	populator.searchSubtree=true

All Authorities Search

These entries populate the BI Platform 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
allAuthoritiesSearch.searchFilter	Filter for "all roles" searches. Active Directory requires that the objectClass value be set to group .	allAuthoritiesSearch.searchFilter=(objectClass

All user name search

These entries populate the BI Platform ACL users.

LDAP Property	Purpose	Example
allUsernamesSearch.usernameAttribute	The attribute used for user values	allUsernamesSearch.usernameAttribute=sAM
allUsernamesSearch.searchBase	Base for "all users" searches	allUsernamesSearch.searchBase=CN=users
allUsernamesSearch.searchFilter	Filter for "all users" searches	allUsernamesSearch.searchFilter=objectClass

Forcing Case-Sensitivity in LDAP

Some LDAP implementations (such as Active Directory) are case-insensitive regarding user names. When using one of these LDAP distributions for your Pentaho authentication backend, when the username typed into the Pentaho User Console login screen doesn't exactly match the case of the user ID in the directory, the directory server will positively authenticate it. However, the user ID that gets passed to the BI Platform contains the exact username that was typed, not the one in the directory. This is a potential security risk. In order to force case sensitivity, follow the below instructions.

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 usernames.

Switching to JDBC

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

Follow the below process to switch from the Pentaho data access object to the JDBC DAO that will allow you to use your own security tables.

 **Note:** If you choose to switch to a JDBC security data access object, you will no longer be able to use the role and user administration settings in the Pentaho Enterprise Console.

1. Stop the BA Server by running the **stop-pentaho** script.
 2. Open the `/pentaho-solutions/system/pentaho-spring-beans.xml` file with a text editor.
 3. Find the following two adjacent lines, and change the **hibernate** to **JDBC** in each:
- ```
<import resource="applicationContext-spring-security-jdbc.xml" />
<import resource="applicationContext-pentaho-security-jdbc.xml" />
```
4. Save the file and close the editor.
  5. Open both of the above-mentioned files and verify that the SQL statements are the correct syntax for your database, and that they reference the correct tables, roles, and actions.
  6. Start the BA Server by running the **start-pentaho** script.

The BI Platform is now configured to authenticate users against the specified database.

## Switching to an LDAP/JDBC Hybrid

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 backend 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 Pentaho Enterprise Console.
2. Open the `/pentaho-solutions/system/pentaho-spring-beans.xml` file with a text editor, find the following two adjacent lines, and change the Spring Security line's **hibernate** segment to **ldap**, and the Pentaho security line's **hibernate** segment to **jdbc**:

```
<import resource="applicationContext-spring-security-ldap.xml" />
<import resource="applicationContext-pentaho-security-ldap.xml" />

<import resource="applicationContext-spring-security-ldap.xml" />
```



```
<import resource="applicationContext-pentaho-security-jdbc.xml" />
```

3. Save and close the file, then open /pentaho-solutions/system/applicationContext-spring-security-ldap.xml file and replace the **populator** bean definition with the one below:

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

4. Remove or comment out the **userDetailsService** bean, then save and close the file.
5. Edit the /pentaho-solutions/system/applicationContext-pentaho-security-jdbc.xml file and add the following two bean definitions, changing the connection and JDBC driver details to match your security database:

```
<bean id="dataSource"
 class="org.springframework.jdbc.datasource.DriverManagerDataSource">
 <property name="driverClassName" value="org.hsqldb.jdbcDriver" />
 <property name="url" value="jdbc:hsqldb:hsq://localhost:9002/userdb" />
 <property name="username" value="sa" />
 <property name="password" value="" />
</bean>
<bean id="userDetailsService"
 class="org.springframework.security.userdetails.jdbc.JdbcDaoImpl">
 <property name="dataSource">
 <ref local="dataSource" />
 </property>
 <property name="authoritiesByUsernameQuery">
 <value> <![CDATA[SELECT username, authority FROM
 granted_authorities WHERE username = ?]]></value>
 </property>
 <property name="usersByUsernameQuery">
 <value> <![CDATA[SELECT username,
 password, enabled FROM users WHERE username = ?]]>
 </value>
 </property>
</bean>
```

6. Edit the /pentaho-solutions/system/data-access/settings.xml file and modify the user and role settings to match your LDAP configuration:

```
<!-- roles with data access permissions -->
<data-access-roles>pentahoAdmins</data-access-roles>
<!-- users with data access permissions -->
<!--
<data-access-users></data-access-users>
-->
<!-- roles with datasource view permissions -->
<data-access-view-roles>pentahoUsers,pentahoAdmins</data-access-view-roles>
<!-- users with datasource view permissions -->
<!-- <data-access-view-users></data-access-view-users> -->
<!-- default view acls for user or role -->
<data-access-default-view-acls>31</data-access-default-view-acls>
```

7. Save and close the file, then edit the /pentaho-solutions/system/applicationContext-pentaho-security-jdbc.xml file and change the connection information and other database-specific details to match your security database:
8. 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 **Admin** and **Authenticated** role values to match the appropriate roles in your JDBC configuration:
  - pentaho.xml
  - repository.spring.xml
  - applicationContext-spring-security.xml
9. Using the command line or your preferred database administration tool, drop the **hibernate.pro\_files** and **hibernate.pro\_acls\_list** columns from the **hibernate** table in your Pentaho solution repository database.
10. Delete the /tomcat/work/ and /tomcat/temp/ directories.
11. Start the BA Server and Pentaho Enterprise Console.
12. Log into the Pentaho Enterprise Console.
13. Configure the Pentaho LDAP connection as explained in [LDAP Properties](#) on page 12.

14. Go to **Configuration > Web Settings**.

15. Under **Authentication** select **LDAP Based** from the drop-down list and click **Submit**.  
The BA Server should restart automatically.

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


## Implementing 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.

### Switching to Central Authentication Service (CAS)

Pentaho can integrate with Central Authentication Service (CAS). Pentaho only supports CAS integration from a manual build and deployment of a Pentaho WAR or EAR file. **If you did not perform a manual deployment of the Pentaho WAR or EAR, or if you deleted your biserver-manual-ee directory, you must retrieve the biserver-manual-ee package from the Pentaho Customer Support Portal.** You must have a CAS server installed and running before you continue. You will also need Apache Ant installed on your system in order to execute the single sign-on script.

Follow the directions below to add single sign-on support (using CAS) to the BI Platform.

 **Important:** This process is **irreversible**, so you should back up your Pentaho **WAR** or **EAR** and the **pentaho-solutions** directory before continuing.

1. After the Pentaho WAR or EAR has been built and deployed, navigate to the `/biserver-manual-ee/build-resources/pentaho-sso/` directory.
2. If your BI Platform or Pentaho Enterprise Console servers are currently running, stop them with the **stop-pentaho** and **stop-pec** scripts, respectively.
3. Edit the **sso-replacements.properties** file and change the default options to match your CAS configuration. Refer to the CAS properties reference below if you have any trouble figuring out what each property does.
4. Use Ant to run the **sso-replacements** script with the **sso-pentaho** switch, as in the following example:

```
ant -f sso-replacements.xml sso-pentaho
```
5. Start the BI Platform and Pentaho Enterprise Console servers with the **start-pentaho** and **start-pec** scripts, respectively.

The BI Platform is now configured to authenticate users against your central authentication server.

### 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.	<code>https://localhost:8443/cas</code>

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

## Switching to Integrated Windows Authentication (IWA)

This procedure requires Microsoft Windows Server 2008 R2, IIS 7.5, and Internet Explorer 8. If you are using different versions of any of this software, you will have to figure out how to adjust the instructions on your own.

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 the directions below to switch to Integrated Windows Authentication in the BA Server.

1. Download the IWA patch JAR from Pentaho: <https://pentaho.box.com/s/0236bea47e7f35abd0e8>.
2. Stop the BA Server, DI Server, and Pentaho Enterprise Console processes.
3. Copy the 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/`
4. In your IIS configuration, disable anonymous authentication and enable Windows authentication for the site you are serving.
5. 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.
6. Edit the `/tomcat/conf/server.xml` file and set **tomcatAuthentication** to **false** in the **Connector** element.



**Note:** If this is not already defined, then add it; the example below can be directly pasted into the file.

```
tomcatAuthentication="false"
```

7. Save and close the file, then edit `/pentaho-solutions/system/applicationContext-spring-security.xml`. Find the **filterChainProxy** filters line, and in it, add **preAuthenticatedProcessingFilter** after **logoutFilter**, separated by a comma.
8. Add the following two bean definitions:



**Note:** This is still in the **applicationContext-spring-security.xml** file, as are the next few steps.

```
<bean id="preAuthenticatedProcessingFilter"
 class="org.pentaho.platform.web.http.security.WindowsPreAuthenticatedProcessingFilter">
 <property name="authenticationManager">
 <ref local="authenticationManager" />
 </property>
</bean>
<bean id="preAuthAuthenticationProvider"

 class=org.springframework.security.providers.preauth.PreAuthenticatedAuthenticationProvide
 <property name="preAuthenticatedUserDetailsService">
 <bean id="userDetailsServiceWrapper"

 class="org.springframework.security.userdetails.UserDetailsByNameServiceWrapper">
 <property name="userDetailsService" ref="userDetailsService" />
 </bean>
 </property>
</bean>
```

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

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

- Replace the **authenticationProcessingFilterEntryPoint** bean definition with the following:

```
<bean id="preAuthenticatedProcessingFilterEntryPoint"

 class="org.springframework.security.ui.preauth.PreAuthenticatedProcessingFilterEntryPoint"
>
```

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

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

- Ensure that you have configured Active Directory integration properly. Refer to your Active Directory documentation and [Microsoft Active Directory Configuration](#) on page 10 for more information.
- Save and close the server.xml file.
- Configure Internet Explorer such that your IIS server is in the **local intranet** security zone.
- Start the BA Server.
- 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.

## Assigning Permissions in the Pentaho User Console

Follow the instructions below to set user role or user account permissions for controlling reports, schedules, subscriptions, and any other content accessible through the Pentaho User Console:

- In the upper left pane, navigate to the location of the content you want to set permissions on. It should appear in the lower left list when you've selected the proper directory.
- In the lower left pane, right-click the item, then click **Properties** in the popup menu.  
A Properties window appears.
- In the Properties window, click the **Advanced** tab.
- Click **Add...** to add a user account or role.  
A new **Select User or Role** window appears.
- In the Select User or Role window, click on the user or role you want to set permissions for, then click **OK**
- In the **Users and Roles** list, click the user or role you want to set permissions for, then click the checkboxes in the **Permissions** list that you want to adjust.
- Repeat the above steps as necessary for other users or roles. When you're finished, click **OK**

The permissions actions you just performed will take effect the next time the specified users log into the Pentaho User Console. See also [Domain Object Authorization](#).

## Permissions Settings

Permission	Effect
All Permissions	Assigns all permissions (explained below) to the specified user or role
Create	Allows a user or role to create reports, analysis views, and dashboards
Update	Allows a user or role to modify an existing report, analysis view, or dashboard
Execute	Allows a user or role to execute or run any content in the solution repository (reports, analysis views, dashboards, but may also include links or other executable content)
Delete	Allows a user or role to delete content from the solution repository
Grant Permission	Allows a user or role to share content with other Pentaho User Console users; essentially, this enables the <b>Share</b> tab in the Properties dialogue as shown in the screen shot below
Schedule	If a user or role has been given access to scheduling functions through the Pentaho Enterprise Console, then this setting in the Pentaho User Console will enable that user or role to arrange for reports or analysis views to be executed at given intervals

### Inheritance

When assigned to a directory, all of the properties listed above will apply to files contained in that directory, including any subdirectories.

# BA Server Content Authorization

---

The BI Platform regulates user- and role-level access to two types of resources: Web Resources and Domain Objects. Web Resources are URLs accessible from the Pentaho User Console; Domain Objects refer to files in the solution repository that make up your BI artifacts (reports, analysis views, dashboards, etc.). This section explains how to modify these access controls according to your preferences.

## User and Role Configuration

---

If you are using the Pentaho security data access object, you can use the Pentaho Enterprise Console to manage users and roles. The subsections below explain how to add, edit, assign, and delete users and roles. Fine-grained permissions are set on file or directory level in the Pentaho User Console.

### Adding Users

You must be logged into the Pentaho Enterprise Console as an administrator user.

To add users in the Pentaho Enterprise Console, follow the directions below.

1. In the Pentaho Enterprise Console, go to the **Administration** section, then click the **Users & Roles** tab.
2. Click the **Users** icon to switch to Users mode.
3. Click the plus sign (+) next to **Users**.  
The **Add Users** dialog box appears.
4. In the **Add Users** dialog box, enter the new user's **User Name**, **Password**, **Password Confirmation** (the same password typed in a second time), and **Description**.
5. Click **OK**.

The specified user account is active, and will appear in the user list.

### Editing User Information

You must be logged into the Pentaho Enterprise Console as an administrator user.

To edit a user account in the Pentaho Enterprise Console, follow the directions below.

1. In the Pentaho Enterprise Console go to the **Administration** section, then click the **Users & Roles** tab.
2. Select the user whose information you want to edit.



**Note:** The user list **Filter** allows you to find specific users in the list. To find a user, type in the first few letters of the user's name in the text box, and a list of names matching your entry appears.

3. In the **Details** pane, edit the user details as needed.
4. Click **Update**.

The changes to the specified user account are immediately applied.

### Deleting Users


You must be logged into the Pentaho Enterprise Console as an administrator user.

To delete in the Pentaho Enterprise Console, follow the directions below.

1. In the Pentaho Enterprise Console, go to the **Administration** section, then click on the **Users & Roles** tab.
2. Select the user or users you want to delete from the Users list.



**Note:** The user list **Filter** allows you to find specific users in the list. To find a user, type in the first few letters of the user's name in the text box, and a list of names matching your entry appears.

3. Click  (**Remove**) next to **Users**.  
The **Delete Users** confirmation dialog box appears.
4. Click **OK** to delete the user(s) and refresh the user list

The specified user accounts are now deleted.

## Adding Roles

You must be logged into the Pentaho Enterprise Console as an administrator user.

To add roles in the Pentaho Enterprise Console, follow the directions below.

1. In the Pentaho Enterprise Console, go to the **Administration** section, then click the **Users & Roles** tab.
2. Click the **Roles** icon to switch to Roles mode.
3. Click the plus sign (+) next to **Roles**.  
The **Add Role** dialog box appears.
4. In the **Add Role** dialog box, enter the **Role Name** and **Description**.
5. Click **OK**  
The new role name appears in the list of roles.

The specified role has been created and is ready to be associated with user accounts.

## Editing Roles

You must be logged into the Pentaho Enterprise Console as an administrator user.

To edit roles in the Pentaho Enterprise Console, follow the directions below.

1. In the Pentaho Enterprise Console, go to the **Administration** section, then click the **Users & Roles** tab.
2. Select the role you want to edit.



**Note:** The user list **Filter** allows you to find specific users in the list. To find a user, type in the first few letters of the user's name in the text box, and a list of names matching your entry appears.

3. In the right pane of the **Roles** page, edit the role **Description** as needed.
4. Click **Update**.

The changes have been applied to the specified role, and will be applied to each user in the group upon their next login.

## Deleting Roles


You must be logged into the Pentaho Enterprise Console as an administrator user.

To delete roles in the Pentaho Enterprise Console, follow the directions below.

1. In the Pentaho Enterprise Console, go to the **Administration** section, then click the **Users & Roles** tab.
2. Click the **Roles** icon if you are not in Roles mode.
3. Select the role or roles you want to delete from the Users list.



**Note:** The user list **Filter** allows you to find specific users in the list. To find a user, type in the first few letters of the user's name in the text box, and a list of names matching your entry appears.

4. Click  (**Remove**) next to **Roles**.  
The **Delete Roles** confirmation dialog box appears.
5. Click **OK** to delete the role(s) and refresh the roles list.

The specified role has been deleted, and any user accounts that had been associated with it will no longer list this role.

## Assigning Users to Roles

You must be logged into the Pentaho Enterprise Console as an administrator user.

To assign users to roles in the Pentaho Enterprise Console, follow the directions below.

1. In the Pentaho Enterprise Console, go to the **Administration** section, then click the **Users & Roles** tab.
2. Click the **Roles** icon to switch to Roles mode.
3. Under **Roles**, select the role that you want to add users to.
4. Click the plus sign (+) next to **Assigned Users**.

The **Assigned Users** dialog box appears.

5. In the **Assigned Users** dialog box, click the arrows to move users in the list under **Available** to (and from) the **Assigned** list.
6. Click **OK**  
Users that have been assigned roles appear in Assigned Users box.

The specified users are now applied to the specified roles. Alternatively, you can assign roles to users in Users mode using a similar process to the one documented above.

## How to Change the Administrator Role

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



**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.

In order for this change to take effect, you will have to [Reapplying the Default Access Control Lists](#) on page 25. This will also reset any administrator permissions you may have set in the Pentaho Enterprise Console.

## Implementing 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 BI Platform 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 BI Platform can now efficiently handle nested roles with LDAP or Active Directory authentication.

Resetting or Creating a new Pentaho Enterprise Console User

The instructions in this section apply to the following situations: (a) You have forgotten the password for accessing the Pentaho Enterprise Console, or, (b) You want to add a new user who will have access the Pentaho Enterprise Console. Follow the instructions below to address either scenario.

1. In Windows, (or Linux), open a command window and go to .\pentaho\server\enterprise-console.  
2. Run the **pec-passwd.bat** file using the parameters below.

Windows	pec-passwd.bat {username} {password}
Linux	pec-passwd.sh {username} {password}

This password utility takes a plain text password as input and returns obfuscated and encrypted versions of the password using various algorithms.

3. Replace {username} with the user name you want to create or change.  
4. Replace {password} with the password you want to generate for the user name that was just created or changed

In the example below the user name is **admin** and the password that is being generated is **password**. Notice that the utility provides you with three different types of encrypted password options: **OBF**, **MD5** and **CRYPT**. Use one of the options in the next step.



```
C:\Program Files\pentaho\server\enterprise-console>pec-passwd.bat admin password
DEBUG: Using PENTAHO_JAVA_HOME
DEBUG: _PENTAHO_JAVA_HOME=C:\Program Files\pentaho\java
DEBUG: _PENTAHO_JAVA=C:\Program Files\pentaho\java\bin\java.exe
DEBUG: PENTAHO_INSTALLED_LICENSE_PATH=C:\PROGRAM~1\pentaho\.installedLicenses.xml
password
OBF:1v2jiuun1xtv1zej1zer1xtn1uoklv1v
MD5:5f4dcc3b5aa765d61d8327deb882cf99
CRYPT:advwtv/9yU5yQ
C:\Program Files\pentaho\server\enterprise-console>
```

5. Edit the `.\pentaho\server\enterprise-console\resource\config\login.properties` file using the information generated by the password encryption utility:



Below is a description of the line in the **login.properties** file (`{username}: {Encryption Type}:{password},{role 1}`):

- `{username}`: the user name used in the password utility in Step 2
- `{Encryption Type}`: the encryption type you are using for your password (OBF, MD5 or CRYPT)
- `{password}`: the encrypted password that was generated using the utility
- `{role}`: the role you want the user to have; there is single role, **admin**, and all users must be granted the admin role.

## Adding Web Resource Authentication

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



**Note:** These instructions are valid across all security DAOs.

1. Ensure that the BI Platform 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.

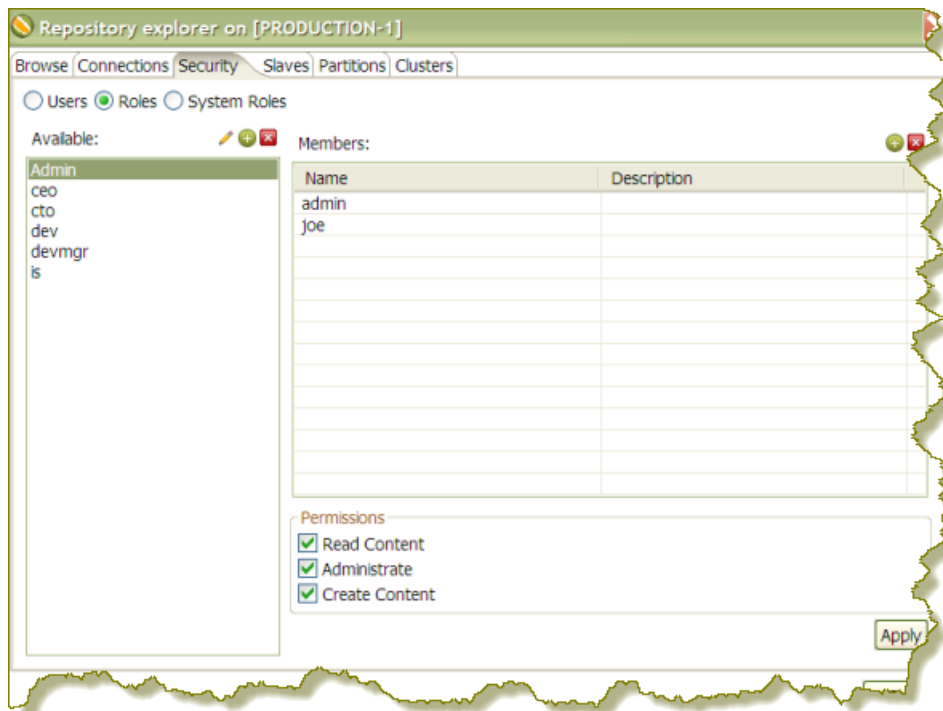
You should now have coarse-grained permissions established for user roles.


## Domain Object Authorization


Domain objects, which are defined in this context as any file in the Pentaho solution repository with a **.url** or **.xaction** extension, can be assigned fine-grained authorization through access control lists (ACLs). Each domain object has one (and only one) ACL, which is created automatically by functions in the Pentaho BI Platform, and may be modified on a user level through the Pentaho User Console.

To establish ACLs, you need to have either created roles and users in the Pentaho Enterprise Console, or successfully connected to your existing security backend. Once a role is established, you can set its permissions in the Pentaho User Console by right-clicking on any file or directory in the file browser, selecting **Properties** from the context menu, then working with the permissions interface, shown below:





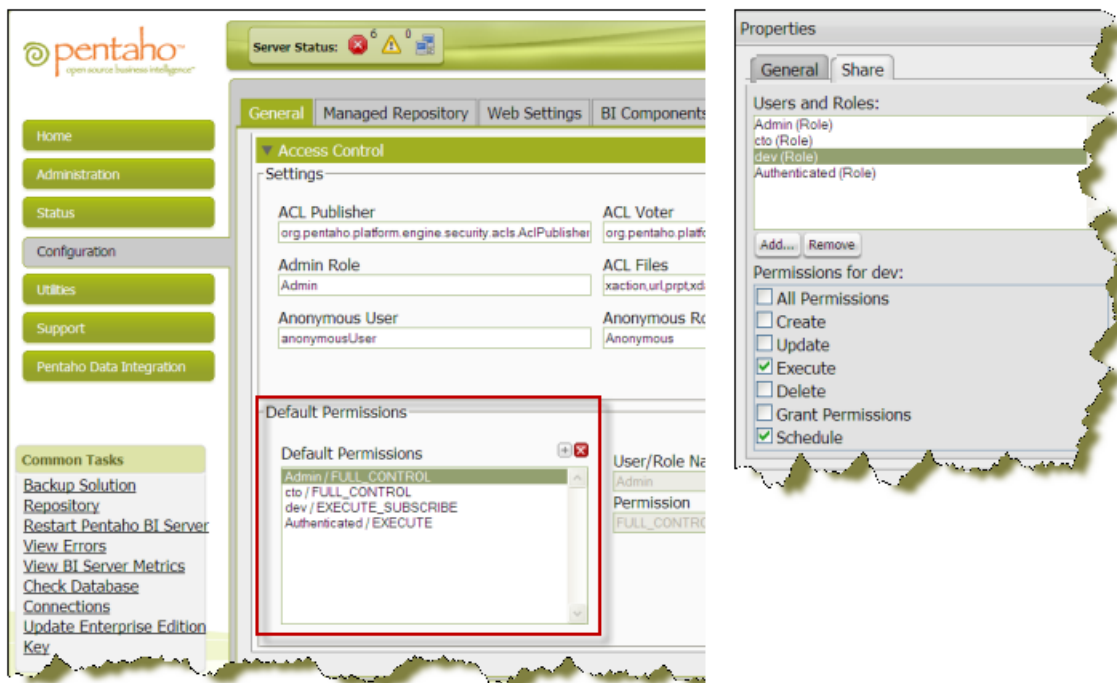
 **Note:** Domain object authorization is only available for database-based solution repositories; file-based solution repositories (an uncommon situation) cannot have this level of fine-grained authorization control.

 **Note:** ACLs are stored in the PRO\_ACL\_FILES table in the solution database, and have no presence in the filesystem. There is, however, a third type of domain object that does not store ACLs in that table -- a metadata model. Metadata models store access controls inline in the metadata model's XMI file. You can define other file extensions to control with access lists by editing the **<acl-files>** entry in the `/pentaho-solutions/system/pentaho.xml` configuration file. See also, [Assigning Permissions in the Pentaho User Console](#).

## Reapplying the Default Access Control Lists

A *default ACL* (labeled **Default Permissions** in the Pentaho Enterprise Console) is a single access control list (ACL) that is applied to all directories in a Pentaho solution repository at the first startup of the BA Server. (The default ACL is copied to each directory — it is not shared among all directories.) After the default ACL has been applied (by starting the server at least once), it can be managed in the Pentaho User Console.

As you examine the image below, notice that users in the *dev* role have execute and scheduling permissions but they cannot create, update, or delete data, nor can they grant permissions to other users. Roles are a way to reference a group of users with a single name. For example, the *Admin* role represents all users who are administrators.



The default ACL will likely not meet your needs, so you may choose to reapply a different default ACL to solution repository objects. To do this, follow the steps below but understand that these steps will remove any ACL management that has been applied in the Pentaho User Console (image on the right).

Follow the process below to reapply the default ACL in the Pentaho Enterprise Console.

1. Log into the Pentaho Enterprise Console.
2. Go to the **Administration** tab.
3. Click **Services**.
4. In the **Solution Repository** tab, click **Restore**.

The access control list will be reapplied with default values you specified.

## Configuring SQL Filters for Dashboards

The Pentaho 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>Admin,DBA</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>Joe,Suzy</sql-execute-users>
```



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

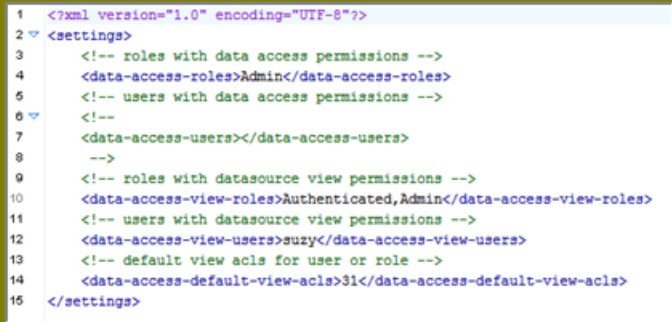
5. Save and close the text editor.
6. Restart the BA Server with the **start-pentaho** script.

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

## Assigning Data Source Permissions for the Pentaho User Console

By default, authenticated users have view-only permissions to data sources in the Pentaho User Console. Users with administrative permissions can create, delete, and view data sources. If you want to fine tune permissions associated with data sources, you must edit the appropriate **settings.xml** file. Follow the instructions below to edit the file.

1. Go to ... \biserver-ee\pentaho-solutions\system\data-access and open **settings.xml**.
2. Edit the settings.xml file as needed. The default values are shown in the sample below. You can assign permissions by individual user or by user role. If you are using LDAP, you can define the correct ACLs value for view permissions; the default value is "31."

A screenshot of a code editor showing the contents of the settings.xml file. The file is an XML document with a root element <settings>. It contains several commented-out sections for roles and users with data access permissions, and one active section for roles and users with datasource view permissions. The active section sets <data-access-view-roles>Authenticated,Admin</data-access-view-roles> and <data-access-view-users>suzy</data-access-view-users>. It also sets a default view ACLs value of 31 in the <data-access-default-view-acls> element.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <settings>
3 <!-- roles with data access permissions -->
4 <data-access-roles>Admin</data-access-roles>
5 <!-- users with data access permissions -->
6 <!--
7 <data-access-users></data-access-users>
8 -->
9 <!-- roles with datasource view permissions -->
10 <data-access-view-roles>Authenticated,Admin</data-access-view-roles>
11 <!-- users with datasource view permissions -->
12 <data-access-view-users>suzy</data-access-view-users>
13 <!-- default view acls for user or role -->
14 <data-access-default-view-acls>31</data-access-default-view-acls>
15 </settings>
```

3. Save your changes to the **settings.xml** file.
4. Restart the Pentaho User Console.

# Securing the Pentaho Enterprise Console and BA Server

This section contains instructions and guidance for enhancing the security of the BA Server and Pentaho Enterprise 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 and/or Pentaho Enterprise Console.

## Configuring SSL (HTTPS) in the Pentaho Enterprise Console and BA Server

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

### Enabling 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 Pentaho Enterprise 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 Pentaho Enterprise 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.

### Enabling 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.

#### Trusting 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 Pentaho Enterprise Console.

1. Change to the home directory of the user account that starts the BA Server and Pentaho Enterprise 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 Pentaho Enterprise 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 someplace 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.

If you'd like to also enable SSL in the Pentaho Enterprise Console, continue on to [Enabling SSL in the Pentaho Enterprise Console](#) on page 29.

## Changing the BA Server Base URL

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

1. Stop the BA Server if it is currently running.  

```
/etc/init.d/pentaho stop
```
2. Navigate to the **WEB-INF** directory inside of the **pentaho.war**.  

```
cd /home/pentaho/pentaho/server/biserver-ee/tomcat/webapps/pentaho/WEB-INF/
```
3. Edit the **web.xml** file.  

```
vim ./web.xml
```
4. Locate the **context-param** element and replace its value to match your new SSL-enabled port number.

```
<context-param>
 <param-name>base-url</param-name>
 <param-value>http://localhost:18443/pentaho/</param-value>
</context-param>
```

Modify the port number in the example to match your configuration.


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

```
/etc/init.d/pentaho start
```

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

## Enabling SSL in the Pentaho Enterprise Console

The information in this section allows you to enable SSL mode in the Pentaho Enterprise Console.

 **Note:** This procedure is optional. You can enable SSL in the BA Server alone, or the Pentaho Enterprise Console alone, or both of them together.

1. Open the **/pentaho/server/enterprise-console-server/resource/config/console.properties** file.  

```
Management Server Enterprise Console's Jetty Server Settings
console.start.port.number=8088
console.hostname=localhost
console.stop.port.number=8033
SSL Section for Enterprise Console
console.ssl.enabled=false
console.ssl.port.number=8143
keyAlias=jetty
keyPassword=changeit
keyStore=resource/config/keystore
keyStorePassword=changeit
trustStore=resource/config/keystore
trustStorePassword=changeit
wantClientAuth=false
needClientAuth=false
Security Authentication Section for Enterprise Console
console.security.enabled=true
console.security.roles.allowed=admin
```

```
console.security.roles.delimiter=,
console.security.realm.name=EnterpriseEdition
console.security.login.module.name=PropertiesFileLoginModule
console.security.auth.config.path=resource/config/login.conf
```

2. Set **console.ssl.enabled** to **true**.
3. Change the values of **keystore** and **trustore** appropriately.
4. Change all of the passwords from the example value of **changeit** to the appropriate values.
5. Change any other values to match your configuration.

The Pentaho Enterprise Console is configured to allow SSL connections.

### Starting the Pentaho Enterprise Console using SSL

To start the console using SSL...

1. Make sure the server is up and running, then open a browser window and type: `https://localhost:8143`. If you are starting the Pentaho Enterprise Console in secure mode for the first time and you have a self-signed certificate, an error message appears.
2. Click the link labeled *Or you can add an exception...*
3. Click **Add Exception**.
4. Click **Get Certificate**.
5. Click **View** if you want to see the details of the certificate.
6. Click **Confirm Security Exception** if you are satisfied with the details of the certificate.  
You are now using an SSL-enabled console.

## Changing Default Enterprise Console Security Settings

The information in this section is based on the Jetty 6.12 and JettyPlus 6.12 release because the Pentaho Enterprise Console uses an embedded Jetty server. Out of the box, the Pentaho Enterprise Console uses a properties based login module but you can plug in any of the login modules listed below or write your own.

- `org.mortbay.jetty.plus.jaas.spi.JDBCLoginModule`
- `org.mortbay.jetty.plus.jaas.spi.PropertyFileLoginModule`

An example of each module is described in the sections that follow but first you must understand the relationship between password handling and LoginModules in the Pentaho Enterprise Console.

Passwords can be stored in clear text, obfuscated, or checksummed. The class, `org.mortbay.util.Password`, must be used to generate all varieties of passwords, the output from which can be cut and pasted into property files or entered into database tables.

 **Important:** Before running **org.mortbay.jetty.security.Password**, you must change directory to enterprise-console. If you do not do a change directory the Jetty JARs will not be found.

```
> cd enterprise-console
> java -cp lib/jetty.jar org.mortbay.jetty.security.Password
Usage - java org.mortbay.util.Password [<user>] <password>
> java -cp lib/jetty.jar org.mortbay.jetty.security.Password me you
you
OBF:20771x1b206z
MD5:639bae9ac6b3e1a84cebb7b403297b79
CRYPT:me/ks90E221EY
```

### JDBCLoginModule

The JDBCLoginModule stores user passwords and roles in a database that are accessed through JDBC calls. You can configure the JDBC connection information, as well as the names of the table and columns storing the user name and credentials, and the name of the table and columns storing the roles.

Below is an example login module configuration file entry for an HSQLDB driver:

```
jdbc {
 org.mortbay.jetty.plus.jaas.spi.JDBCLoginModule required
 debug="true"
```

```

dbUrl="jdbc:hsqldb:."
dbUserName="sa"
dbDriver="org.hsqldb.jdbcDriver"
userTable="myusers"
userField="myuser"
credentialField="mypassword"
userRoleTable="myuserroles"
userRoleUserField="myuser"
userRoleRoleField="myrole";
};

```


There is no particular schema required for the database tables storing the authentication and role information. The properties `userTable`, `userField`, `credentialField`, `userRoleTable`, `userRoleUserField`, `userRoleRoleField` configure the names of the tables and the columns within them that are used to format the following queries:

```

select <credentialField> from <userTable> where <userField> =?
 select <userRoleRoleField> from <userRoleTable> where <userRoleUserField>
=?

```

Credential and role information is read from the database when a previously unauthenticated user requests authentication. Note that this information is only cached for the length of the authenticated session. When the user logs out or the session expires, the information is flushed from memory.

 **Note:** Passwords can be stored in the database in plain text or encoded formats, using the `org.mortbay.jetty.security.Password` class.

## PropertyFileLoginModule

With this login module implementation, the authentication and role information is read from a property file:

```

props {
 org.mortbay.jetty.plus.jaas.spi.PropertyFileLoginModule required
 debug="true"
 file="/somewhere/somefile.props";
};

```

The file parameter is the location of a properties file of the same format as the `etc/realm.properties` example file as shown below:

```
<username>: <password>[<rolename> ...]
```

Below is an example:

```

admin: OBF:1xmk1w261u9rlw1clxm9,user,admin
superadmin: changeme,user,developer
master: MD5:164c88b302622e17050af52c89945d44,user
: CRYPT:adpexzg3FUZAk,admin

```

The contents of the file are read and cached in memory the first time a user requests authentication.

## Editing Security Settings

Security settings configuration are stored in the security section of `console.properties` file:

```

Management Server Enterprise Console's Jetty Server Settings
console.start.port.number=8088
console.hostname=localhost
console.stop.port.number=8033

SSL Section for Enterprise Console
console.ssl.enabled=false
console.ssl.port.number=8143
keyAlias=jetty
keyPassword=changeit
keyStore=resource/config/keystore
keyStorePassword=changeit
trustStore=resource/config/keystore
trustStorePassword=changeit
wantClientAuth=false
needClientAuth=false

```



```
Security Authentication Section for Enterprise Console
console.security.enabled=true
console.security.roles.allowed=admin
console.security.roles.delimiter=,
console.security.realm.name=EnterpriseEdition
console.security.login.module.name=PropertiesFileLoginModule
console.security.auth.config.path=resource/config/login.conf
```

Security is enabled by default.


- To change the roles you want to allow the application to access you must provide your list of roles in **console.security.roles.allowed property**.
- Roles are comma separated by default; you can change that configuration by providing your delimiter in **console.security.roles.delimiter property**.
- The login module name must be provided for the property name, **console.security.login.module.name**. This is the name you gave your login module in the **login.conf** file.
- You must provide the location of your **login.conf** file in **console.security.auth.config.path property**.

## Changing the Admin Credentials for the Pentaho Enterprise Console

The default user name and password for the Pentaho Enterprise Console are **admin** and **password**, respectively. You must change these credentials if you are deploying Pentaho in a production environment. Follow the instructions below to change credentials.

 **Important:** Before running **org.mortbay.jetty.security.Password**, you must change directory to **enterprise-console**. If you do not do a change directory the Jetty JARs will not be found.

1. Generate the password from the command line.

 **Note:** You may need to change version numbers in the file names. The example below is for Linux:

```
$ cd enterprise-console
$ java -cp lib/jetty-6.1.2rc1.jar:lib/jetty-util-6.1.2rc1.jar
org.mortbay.jetty.security.Password username password
password
OBF:1v2jluumlxtvlzejlzerlxtnluvklv1v
MD5:5f4dcc3b5aa765d61d8327deb882cf99
CRYPT:usjRS48E8ZADM
```

For Windows use:

```
$ cd enterprise-console
$ java -cp lib/jetty-6.1.2rc1.jar;lib/jetty-util-6.1.2rc1.jar
org.mortbay.jetty.security.Password username password
```

2. Go to ... \enterprise-console\resource\config and open the **login.properties** file.
3. Edit the file using the following format:

```
<username>: OBF:<obfuscated_password>,<role1>,<role2>,<role3>,...
```

## Creating a Custom Login Module

To create a custom login module you must be familiar with the following java classes, **AbstractLoginModule.java** and **UserInfo.java**, shown below:

```
package org.mortbay.jetty.plus.jaas.spi; public abstract class
 AbstractLoginModule implements LoginModule { ... public abstract
 UserInfo
 getUserInfo (String username) throws Exception; }

package org.mortbay.jetty.plus.jaas.spi; public class UserInfo { public
 UserInfo (String userName, Credential credential, List roleNames) { ... }
 public String
 getUserName() { ... } public List getRoleNames () { ... } public boolean
 checkCredential
 (Object suppliedCredential) { ... } }
```



The `org.mortbay.jetty.plus.jaas.spi.AbstractLoginModule` implements all of the `javax.security.auth.spi.LoginModule` methods. All you must do is implement the `getUserInfo` method to return a `org.mortbay.jetty.plus.jaas.UserInfo` instance that encapsulates the user name, password and role names (as `{java.lang.String}s`) for a user.



**Note:** The `AbstractLoginModule` does not support caching. If you want to cache `UserInfo` (for example, as does the `org.mortbay.jetty.plus.jaas.spi.PropertyFileLoginModule`) you must provide it yourself.

## Using 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 *Pentaho Performance-Tuning Guide*. If you have already followed the Apache httpd procedure in that guide, there is no need to perform it again with the instructions below.

## Using 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.

# Removing Security

You cannot remove the security mechanisms built into the BI Platform, but you can bypass them by giving all permissions to anonymous users. The BA Server automatically assigns all unauthenticated users to the **anonymousUser** account and the **Anonymous** role. The procedure below will grant full BA Server access to the Anonymous role, thereby never requiring a login.



**Danger: This process is irreversible!** While you can simply back out any configuration file changes you may make, you cannot restore custom ACLs once they've been wiped out; the best you can do is restore the default ACLs and make all of your custom authorization changes by hand again. Before performing the below procedure, you should have a very good reason for bypassing security.

## 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. The username does not matter in this particular bean; only the role name.

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

## 3. Now find the **filterSecurityInterceptor** bean in the same file, and the **objectDefinitionSource** property inside of it, and match its contents to the example below:

This step allows Pentaho client tools to publish to the BI Platform without having to supply a username and password.

```
<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>
```

## 4. Save the file, then open `pentahoObjects-spring.xml` in the same directory.

## 5. Change the **IAclVoter** class to **PentahoAllowAnonymousAclVoter**

```
<beans>
<!-- omitted -->
 <bean id="IAclVoter" class="org.pentaho.platform.engine.security.acls
.voter.PentahoAllowAnonymousAclVoter" scope="singleton" />
<!-- omitted -->
</beans>
```

## 6. Save the file, then open `pentaho.xml` in the same directory.

## 7. In the `<anonymous-authentication>` part of the **<pentaho-system>** section, define the anonymous user and role.

This is the same user and role you will use when assigning ACLs in the next step.

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

- Using the same anonymous user and role from before, adjust the ACLs accordingly and remove all ACL overrides.

```
<pentaho-system>
<!-- omitted -->
 <acl-publisher>
 <default-acls>
 <acl-entry role="Anonymous" acl="ADMIN_ALL" />
 <acl-entry role="Authenticated" acl="ADMIN_ALL" /> </default-acls>
 <!-- remove any active overrides entries -->
 </acl-publisher>
 <!-- omitted -->
</pentaho-system>
```

- Adjust the <acl-voter> properties such that the new anonymous user has administrator privileges.

```
<pentaho-system> <!-- omitted -->
 <acl-voter>
 <admin-role>Anonymous</admin-role>
 </acl-voter> <!-- omitted -->
</pentaho-system>
```

- Save the file and close the text editor.

You've now effectively worked around the security features of the BI Platform.

You should now follow the procedures below to remove the security awareness from the metadata domain repository, and switch from a database repository to a file repository; database repositories are only advantageous when you need to assign ACLs to certain resources. Since you no longer need to do that, you can eliminate your solution database to remove the last piece of BI Platform authorization security.

## Switching the Metadata Domain Repository

The Pentaho BA Server and Pentaho Enterprise Console must be stopped before executing these instructions.

This procedure changes your default metadata domain repository so that it is no longer security-aware. It is a necessary step in completely removing security from the BI Platform; however, this procedure does not, in itself, remove **all** security. To do that, start with [Removing Security](#) on page 36.

- Edit the /pentaho-solutions/system/pentahoObjects.spring.xml file.
- Comment out the **IMetadataDomainRepository** line, and uncomment the similar line below it.

Alternatively, you can switch the value of **IMetadataDomainRepository** from **org.pentaho.platform.plugin.services.metadata.SecurityAwareMetadataDomainRepository** to **org.pentaho.platform.plugin.services.metadata.MetadataDomainRepository**.

```
<!-- <bean id="IMetadataDomainRepository"
class="org.pentaho.platform.plugin.services.metadata.SecurityAwareMetadataDomainRepository"
scope="singleton"/> -->
<!-- Use this schema factory to disable PMD security -->
<bean id="IMetadataDomainRepository"
class="org.pentaho.platform.plugin.services.metadata.MetadataDomainRepository"
scope="singleton"/>
```

- Save and close the file.

You've now switched the metadata domain repository to one that is not security-aware. Access controls will no longer be enforced on various metadata objects.

You should now continue on to [Switching to a File-Based Solution Repository](#) on page 37.

## Switching to a File-Based Solution Repository

The Pentaho BA Server and Pentaho Enterprise Console must be stopped before executing these instructions.

This procedure changes your default database solution repository into a strictly file-based repository. There will no longer be a database mirroring the content in your pentaho-solutions directory, and there will be no way of isolating or securing BI content within the BA Server. However, removing the database overhead means that there will be a substantial performance increase in many instances. This procedure does not, in itself, remove security from the BI Platform, but it does remove access control lists from all content.

1. Edit the `/pentaho-solutions/system/pentahoObjects.spring.xml` file.
2. Comment out the current **`ISolutionRepository`** line, and uncomment the similar line above it.

Alternatively, you can switch the value of **`ISolutionRepository`** from **`org.pentaho.platform.repository.solution.dbbased.DbBasedSolutionRepository`** to **`org.pentaho.platform.repository.solution.filebased.FileBasedSolutionRepository`**.

```
<!-- Uncomment the following line to use a filesystem-based repository. Note: does not
support ACLs. -->
<bean id="ISolutionRepository"
 class="org.pentaho.platform.repository.solution.filebased.FileBasedSolutionRepository"
 scope="session" />
<!-- Uncomment the following line to use a filesystem/db-based repository with meta
information stored in a db -->
<!-- <bean id="ISolutionRepository"
 class="org.pentaho.platform.repository.solution.dbbased.DbBasedSolutionRepository"
 scope="session" /> -->
```

3. Comment out the **`IMetadataDomainRepository`** line, and uncomment the similar line below it.

Alternatively, you can switch the value of **`IMetadataDomainRepository`** from **`org.pentaho.platform.plugin.services.metadata.SecurityAwareMetadataDomainRepository`** to **`org.pentaho.platform.plugin.services.metadata.MetadataDomainRepository`**.

```
<!-- <bean id="IMetadataDomainRepository"
 class="org.pentaho.platform.plugin.services.metadata.SecurityAwareMetadataDomainRepository"
 scope="singleton"/> -->
<!-- Use this schema factory to disable PMD security -->
<bean id="IMetadataDomainRepository"
 class="org.pentaho.platform.plugin.services.metadata.MetadataDomainRepository"
 scope="singleton"/>
```

4. Save and close the file.

You've now switched over to a file-based solution repository. You can safely restart your BA Server and Pentaho Enterprise Console server.

# Metadata Security

---

Most of the security configuration explained thus far has involved using features that are built into the Pentaho User Console or Pentaho Enterprise Console, or involve configuration changes of some kind. However, if you need to restrict access to certain portions of a metadata model that you are using as a data source, the only way to do it is to edit the model with Metadata Editor and add restrictions. The Pentaho metadata model offers table-, column-, and row-level authorization control, so if you need to prevent certain users or roles from accessing it, you can change (and republish) the model accordingly.

## Configuring the Security Service

---

You must know the base URL for the Pentaho BA Server (the default URL is **`http://localhost:8080/pentaho`** as well as the name of the service to execute security information retrieval (the service is **ServiceAction**).

The Pentaho Metadata Editor must be configured to connect to your BA Server so that it can retrieve usernames, roles, and access control lists. Follow the below directions to set up Metadata Editor.

1. Start the Pentaho Metadata Editor.

```
sh /pentaho/design-tools/metadata-editor/metaeditor.sh
```

2. Go to the **Tools** menu, then select **Security...**

The Security Service dialogue will appear.

3. In the **Service URL** field, type in the base URL for the BA Server plus the security service.

```
http://localhost:8080/pentaho/ServiceAction
```

4. Next, select the level of detailed security information you want: **All**, **Users** or **Roles**.

If you have hundreds of users in your system, you probably only want to return the roles, and use roles for security information properties. The access control lists are returned with all three options.

5. In the **Username** and **Password** fields, type in the login credentials for an administrator-level account.

6. Click **Test**.

A popup window with the returned XML should appear. If it does not, check that the information you typed into the fields mentioned above is correct.

## Adding Column-Level Security Constraints

---

You need to have established a connection to a data source in Metadata Editor and selected one or more tables to create metadata for.

Follow the below instructions to add user- or role-based restrictions to your data source.

1. In the left pane, right-click the table or column you want to modify, then click **Edit..** from the context menu.

The **Physical Table Properties** dialogue will appear.

2. Click the green **+** icon above the **Available** field in the middle of the screen.

The **Add New Property** dialogue will appear.

3. Select **Metadata Security**, then click **OK**.

4. Click the new **Metadata Security** item in the **General** category.

5. Click the green **+** icon next to the **Selected Users/Groups** field in the right pane.

A list of users and/or roles (depending on what you selected when configuring the security service earlier) will appear.

6. Click the user or role in the left pane that you want to assign permissions to, then click the right arrow button in the middle of the window.

The user or role will move from the **Available** list on the left to the **Assigned** list on the right.

7. Click the checkboxes for the permissions that you want to assign to the selected user or role.

8. Repeat this process for other users or roles you want to assign metadata permissions to, then click **OK**.

9. Change any other relevant metadata options, then click **OK** to return to the Metadata Editor main window.

10. When you are finished, save the metadata configuration as a domain using the **Save As** button, then publish it to the BA Server as an XML schema by selecting **Publish** from the **File** menu.

## Adding Global Row-Level Security Constraints

You need to have established a connection to a data source in Metadata Editor and selected one or more tables to create metadata for.

A global constraint is an MQL Formula statement that institutes a global restriction on the data you specify, down to the row level. Follow the below instructions to add custom global user- or role-based restrictions to your data source.

1. In the left pane, right-click the table or column you want to modify, then click **Edit..** from the context menu.  
The **Physical Table Properties** dialogue will appear.
2. Click the green **+** icon above the **Available** field in the middle of the screen.  
The **Add New Property** dialogue will appear.
3. Select **Data Constraints**, then click **OK**.
4. Click the new **Data Constraints** item in the **General** category.
5. Select the **Global Constraint** option in the right pane.
6. Type in your constraint in the provided text box.
7. Change any other relevant metadata options, then click **OK** to return to the Metadata Editor main window.
8. When you are finished, save the metadata configuration as a domain using the **Save As** button, then publish it to the BA Server as an XML schema by selecting **Publish** from the **File** menu.

## MQL Formula Syntax For Global Constraints

You can use all of the standard operators, and any of the following functions when defining a global constraint:

Function Name	Parameters	Description
OR	Two or more boolean expressions	Returns <b>true</b> if one or more parameters are true
AND	Two or more boolean expressions	Returns <b>true</b> if all parameters are true
LIKE	Two	Compares a column to a regular expression, using % as a wild card
IN	Two or more	Checks to see if the first parameter is in the following list of parameters
NOW	N/A	The current date
DATE	Three numeric parameters: Year, month, and day	The specified date
DATEVALUE	One text parameter: year-month-day	The specified date
CASE	Two or more	Evaluates the odd-numbered parameters, and returns the even numbered parameter values. If there are an odd number of parameters, the last parameter is returned if no other parameter evaluates to true.
COALESCE	One or more	Returns the first non-null parameter. If all are null, the message in the last parameter is returned.
DATEMATH	One expression	Returns a date value based on a DATEMATH expression (see related links below for a link to the DATEMATH Javadoc)

### OR

```
OR([BT_CUSTOMERS.BC_CUSTOMERS_CUSTOMENAME] = "EuroCars";
 [BT_CUSTOMERS.BC_CUSTOMERS_CREDITLIMIT] > 1000)
```



**AND**

```
AND([BT_CUSTOMERS.BC_CUSTOMERS_CUSTOMENAME] = "EuroCars";
 [BT_CUSTOMERS.BC_CUSTOMERS_CREDITLIMIT] > 1000)
```

**LIKE**

```
LIKE([BT_CUSTOMERS.BC_CUSTOMERS_CUSTOMENAME]; "%SMITH%")
```

**IN**

```
IN([BT_CUSTOMERS.BC_CUSTOMERS_CUSTOMENAME]; "Adam Smith"; "Brian Jones")
```

**NOW**

```
NOW()
```

**DATE**

```
DATE(2008;4;15)
```

**DATEVALUE**

```
DATEVALUE("2008-04-15")
```

**CASE**

```
CASE([BT_CUSTOMERS.BC_CUSTOMERS_CUSTOMENAME] = "EuroCars"; "European
Cars";
 [BT_CUSTOMERS.BC_CUSTOMERS_CUSTOMENAME] = "AsiaCars";
 "Asian Cars"; "Unknown Cars"
)
```

**COALESCE**

```
COALESCE([BT_CUSTOMERS.BC_CUSTOMERS_CUSTOMENAME];
 [BT_CUSTOMERS.BC_CUSTOMERS_CUSTOMERID]; "Customer is
Null")
```

**DATEMATH**

```
DATEMATH("0:ME -1:DS")
```

This expression represents 00:00:00.000 on the day before the last day of the current month.

## Adding User or Role Row-Level Security Constraints

You need to have established a connection to a data source in Metadata Editor and selected one or more tables to create metadata for.

A role-based constraint is an MQL Formula statement that restricts access (on the row level) only to certain users or roles. Follow the below instructions to add fine-grained user- or role-based restrictions to your data source.

1. In the left pane, right-click the table or column you want to modify, then click **Edit..** from the context menu.  
The **Physical Table Properties** dialogue will appear.
2. Click the green **+** icon above the **Available** field in the middle of the screen.

The **Add New Property** dialogue will appear.

3. Select **Data Constraints**, then click **OK**.
4. Click the new **Data Constraints** item in the **General** category.
5. Select **Role-Based Constraints** option in the right pane.
6. Click the green **+** icon next to the **Selected Users/Groups** field in the right pane.  
A list of users and/or roles (depending on what you selected when configuring the security service earlier) will appear.
7. Click the user or role in the left pane that you want to assign permissions to, then click the right arrow button in the middle of the window.  
The user or role will move from the **Available** list on the left to the **Assigned** list on the right.
8. Click the checkboxes for the permissions that you want to assign to the selected user or role.
9. Repeat this process for other users or roles you want to assign metadata permissions to, then click **OK**.
10. Change any other relevant metadata options, then click **OK** to return to the Metadata Editor main window.
11. When you are finished, save the metadata configuration as a domain using the **Save As** button, then publish it to the BA Server as an XML schema by selecting **Publish** from the **File** menu.

## MQL Formula Syntax For User and Role Row-Level Constraints

The MQL Formula syntax for defining a user or role row constraint is:

```
[table.column] = "row"
```

The table and column are defined as part of a metadata business model. Here is an example that isolates access to data from the Sales department:

```
[BT_OFFICE.BC_DEPARTMENT] = "Sales"
```

It's also possible to give or deny access to an entire role, or a single user, by selecting that user or role, then using a boolean for a constraint:


```
TRUE()
```

Or:

```
FALSE()
```

## Restricting Metadata Models to Specific Client Tools

To prevent a published metadata model from being displayed in the data source lists of one or more Pentaho User Console client tools (Dashboard Designer, Interactive Reporting, and ad hoc reporting), follow this process:

 **Note:** ROLAP data sources for Analyzer and JPivot are not affected by this configuration.

1. Start Metadata Editor and open the metadata model you want to restrict.
2. Right-click on the business model, then select **Edit** from the context menu.  
The **Business Model Properties** window will appear.
3. Click the round green **+** icon to add a new custom property.  
The **Add New Property** window will appear.
4. Click **Add a custom property**, then type **visible** in the **ID** field and select **String** from the **Type** drop-down box, then click **OK**.  
You'll return to the **Business Model Properties** window.
5. Scroll down to the bottom of the window to see the **Value** field. Type in one or more of the following restriction values, separated by commas, then click **OK**:
  - **adhoc**: removes the model from the list in ad hoc reporting
  - **dashboard-filters**: prevents this model from appearing in dashboard filters that use metadata queries
  - **dashboard-content**: prevents this model from being used in dashboard panels
  - **interactive-reporting**: removes the model from the list in Interactive Reporting

- **manage**: removes this model from the "manage data sources" feature of the Pentaho User Console



**Note:** If you leave the **Value** field blank, the published model will not appear in any Pentaho User Console tools.

6. Re-publish this model to the BA Server.
7. If the BA Server is currently running, restart it.

The restrictions you defined should now be in place.

# Analysis Schema Security

You can restrict portions of your ROLAP schema from being viewed by certain user roles defined within your schema definition, and then map those role restrictions to the BA Server so that when you publish the schema, its security restrictions are obeyed in Analyzer and Dashboard Designer (or any other BA Server-based dashboard). If you do this, only the permissible parts of the schema will be available to the specified roles. Pentaho offers three unique paths to accomplish role-based security, all of which are explained later in this section.

If you need more granularity than role-level restrictions can provide, you can alternatively create a special ROLAP schema for each different user and change the BA Server configuration so that each Pentaho User Console user has access to their own private schema. Refer to the subsections below that apply to your situation.



**Note:** Changes to a ROLAP schema should be done with Schema Workbench. If you change a schema, you must republish it to the BA Server in order for the modifications to take effect. Changes to XML configuration files (such as those that contain core BA Server and Mondrian engine settings or properties) can be done with any text editor.

## Restricting Access to Specific Members

You can restrict access to parts of a schema by implementing the **<HierarchyGrant>** element to define access to a hierarchy:

```
<Role name="California manager">
 <SchemaGrant access="none">
 <CubeGrant cube="Sales" access="all">
 <HierarchyGrant hierarchy="[Store]" access="custom" topLevel="[Store].[Store
Country]">
 <MemberGrant member="[Store].[USA].[CA]" access="all"/>
 <MemberGrant member="[Store].[USA].[CA].[Los Angeles]" access="none"/>
 </HierarchyGrant>
 <HierarchyGrant hierarchy="[Customers]" access="custom"
topLevel="[Customers].[State Province]" bottomLevel="[Customers].[City]">
 <MemberGrant member="[Customers].[USA].[CA]" access="all"/>
 <MemberGrant member="[Customers].[USA].[CA].[Los Angeles]"
access="none"/>
 </HierarchyGrant>
 <HierarchyGrant hierarchy="[Gender]" access="none"/>
 </CubeGrant>
 </SchemaGrant>
</Role>
```

The **access** attribute can be **"all"**, meaning all members are visible; **"none"**, meaning the hierarchy's very existence is hidden from the user; and **"custom"**. With custom access, you can use the **topLevel** attribute to define the highest visible level (preventing users from seeing too much of the 'big picture,' such as viewing revenues rolled up to the Store or Country level); or use the **bottomLevel** attribute to define the lowest visible level (preventing users from looking at an individual customer's details); or control which sets of members a user can see by defining nested **<MemberGrant>** elements.

You can only define a **<MemberGrant>** element if its enclosing **<HierarchyGrant>** has **access="custom"**. Member grants give (or remove) access to a given member and all of its children. Here are the rules:

- 1. Members inherit access from their parents.** If you deny access to California, you won't be able to see San Francisco.
- 2. Grants are order-dependent.** If you grant access to USA, then deny access to Oregon, then you won't be able to see Oregon or Portland. But if you were to deny access to Oregon, then grant access to USA, you can effectively see everything.
- 3. A member is visible if any of its children are visible.** Suppose you deny access to USA, then grant access to California. You will be able to see USA, and California, but none of the other states. The totals against USA will still reflect all states, however.
- 4. Member grants don't override the hierarchy grant's top- and bottom-levels.** If you set **topLevel="[Store].[Store State]"**, and grant access to California, you won't be able to see USA.

## Mondrian Role Mapping in the BA Server

The role mapper connects user role restrictions defined in a Mondrian schema to user roles defined in the Pentaho BA Server. This enables BI developers to set schema access controls in a single place, rather than in many places across different parts of Business Analytics. If you do not configure the role mapper, then none of the roles defined in your schema will restrict access in the BA Server.

The role mapper is configured through the BA Server in the `/pentaho-solutions/system/pentahoObjects.spring.xml` file. There are three mapper implementations available, each with disabled example configurations in `pentahoObjects.spring.xml`. All three are explained in the below subsections.



**Note:** When a role is defined in an action sequence, it will override the Mondrian role mapper.

### The Mondrian One-To-One UserRoleMapper

The **Mondrian-One-To-One-UserRoleMapper** maps each role name in the BA Server to roles defined in the ROLAP schema. Therefore, the mapper assumes that the roles defined in your ROLAP schema are mirrored in the BA Server. For example, if you have a role called "CTO" in your schema, and a role called "CTO" in the BA Server, this role mapper would be appropriate.

```
<bean id="Mondrian-UserRoleMapper"
name="Mondrian-One-To-One-UserRoleMapper"
class="org.pentaho.platform.plugin.action.mondrian.mapper
.MondrianOneToOneUserRoleListMapper"
scope="singleton" />
```

### The Mondrian SampleLookupMap UserRoleMapper

This mapper provides a translation table (in the form of a **<map>** element) to associate BA Server roles with ROLAP schema roles. The lookups take the form of key/value pairs where the key is the BA Server role, and the value is the ROLAP schema role. In the example below, the "ceo" role in the BA Server maps to the "California manager" role in the schema.

```
<bean id="Mondrian-UserRoleMapper"
 name="Mondrian-SampleLookupMap-UserRoleMapper"
 class="org.pentaho.platform.plugin.action.mondrian.mapper.
 MondrianLookupMapUserRoleListMapper"
 scope="singleton">
 <property name="lookupMap">
 <map>
 <entry key="ceo" value="California manager" />
 <entry key="cto" value="M_CTO" />
 <entry key="dev" value="M_DEV" />
 </map>
 </property>
</bean>
```

### The Mondrian SampleUserSession UserRoleMapper

This mapper retrieves ROLAP schema roles from a named HTTP session variable. In the below example, the session is stored in a variable called **MondrianUserRoles**.

```
<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>
```

## Restricting ROLAP Schemas Per User

To enable user-level access restrictions for each published analysis schema in the Pentaho User Console, follow the instructions below.



**Note:** There is a different process for restricting ROLAP schemas per role. See [Mondrian Role Mapping in the BA Server](#) for instructions on role mapping.

1. Log into the Pentaho User Console as the administrator user.
2. Create new a solution directory for the user you want to provide a private analysis schema for.
3. Change the access permissions on the new user directory so that only the specified user has access to it.
4. Copy the target schema to the private directory that you just created.
5. Rename the copied schema such that it reflects the user account that now owns it.
6. Refresh the solution repository.
7. Edit the `/pentaho/server/biserver-ee/pentaho-solutions/system/olap/datasources.xml` file and add data source entries for each personalized schema copy you created.

```
<Catalog name="Suzy">
 <DataSourceInfo>Provider=mondrian;DataSource=SampleData</DataSourceInfo>
 <Definition>solution:suzy/suzy.mondrian.xml</Definition>
</Catalog>
<Catalog name="Tiffany">
 <DataSourceInfo>Provider=mondrian;DataSource=SampleData</DataSourceInfo>
 <Definition>solution:tiffany/tiffany.mondrian.xml</Definition>
</Catalog>
```

8. Edit the `/pentaho/server/biserver-ee/pentaho-solutions/system/pentaho.xml` file and add a `,xml` to the end of the **acl-files** list.

```
<acl-files>xaction,url,prpt,xdash,xcdf,xanalyzer,xanalyzer,xml</acl-files>
```

9. Restart the BA Server.

The schema copies you created are now only available to the users you specified.

# Using Security Information In Action Sequences

In addition to providing access to security information within the Web application code, the BI Platform's security system also provides access to security information within action sequences. That security information then can be used in JavaScript rules, presented in reporting prompts, provided as input to SQL lookup rules, etc.

A typical action sequence inputs section is defined like this:

```
<inputs>
 <someInput type="string">
 <sources>
 <request>someInput</request>
 </sources>
 </someInput>
</inputs>
```

In the above example, the input (called **someInput**) can be found by looking at the request (HttpServletRequest, PortletRequest, etc.) for a variable called **someInput**. Then, throughout the rest of the action sequence, specific actions can reference that input. The Pentaho BI Platform extends the inputs to provide a unique type of input: The security input. This input presently supports the following input names:

Input Name	Type	Description
principalName	string	The name of the currently authenticated user.
principalRoles	string-list	The roles that the currently authenticated user is a member of.
principalAuthenticated	string	true if the user is authenticated, false otherwise.
principalAdministrator	string	true if the user is considered a Pentaho Administrator, false otherwise.
systemRoleNames	string-list	All the known roles in the system. Use caution since this list could be quite long.
systemUserNames	string-list	All the users known to the system. Use caution since this list could be quite long.

The following input section will get the list of the user's roles, and make it available to all the actions in the action sequence:

```
<inputs>
 <principalRoles type="string-list">
 <sources>
 <security>principalRoles</security>
 </sources>
 </principalRoles>
</inputs>
```

# 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 a different security DAO 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.

## Miscellaneous Troubleshooting Tips

---

### **sun.security.validator.ValidatorException: No trusted certificate found**

This error occurs when you are trying to enable SSL to securely connect to the Pentaho Enterprise Console, and have not executed the procedure to trust your SSL certificate. Follow these instructions: [Enabling SSL in the Pentaho Enterprise Console](#) on page 29

## Increasing Security Log Levels in the BI Platform

---

The security logging facilities of the BI Platform 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 BI Platform 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 backend 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`

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

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


8. Save the file and close the text editor.



## 9. Start the BA Server.


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

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

 **Note:** When you are finished configuring and testing the BI Platform, you should adjust the log levels down to a less verbose level, such as ERROR, to prevent `pentaho.log` from growing too large.

## Enabling 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. Usernames 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>
 <bean id="loggingAdvisor"
 class="org.springframework.aop.support.RegexpMethodPointcutAdvisor">
 <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 username and/or password doesn't match what's stored in the backend, 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 username and password match, you should see a log message that looks like the following example. 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"

At this stage, you should be pretty much ready to go as long as the roles that are in your AD are Admin and Authenticated... Hands up all those LDAP administrators that have those two roles in their ldap – Your Fired! You do not to be using those roles in your LDAP. So I will now show you how to configure your system to use the roles: pentahoAdmins and pentahoUsers. These can be anything, I have chosen these roles to be easily identifiable. The file you want to be editing at this point is the `/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 what is known as URL Security. They are regular expressions to match a path on the browser's URL and require the user to be a member of the defined role to gain access. In the example above, both anonymous and Authenticated get access. Now, as specified above, we don't want Authenticated, we want pentahoUsers. Well, let's change it. `A/docs/. *Z=Anonymous,pentahoUsers` Now do the same for every other entry here. Where you see Authenticated, replace it with pentahoUsers (or whatever you choose for Users). Where you see Admin, replace it with pentahoAdmins (or whatever you choose for Admins). For Authenticated to pentahoUsers you can safely 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 an enterprise repository from Spoon before you switch the authentication backend to LDAP, then the repository IDs and security structures will be broken. You won't see an error message, but the enterprise 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 the LDAP backend.



**Danger:** Following this procedure will destroy any previously defined enterprise repository users, roles, and access controls. You should back up the files that you're instructed to delete below.

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 enterprise repository that can store content and create new directories.

## LDAP incorrectly authenticates user IDs that don't 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 backend, you might run into an issue where a valid username with invalid letter cases will improperly validate. For instance, if **Bill** is the valid user ID, and someone types in **bill** at the Pentaho 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: [Forcing Case-Sensitivity in LDAP](#) on page 13.