Version 5.3

Define DI Server Advanced Security

Copyright Page

Help and Support Resources

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

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

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

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

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

Liability Limits and Warranty Disclaimer

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

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

Trademarks

Third-Party Open Source Software

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

Contact Us

Global Headquarters Pentaho Corporation Citadel International, Suite 460

5950 Hazeltine National Drive Orlando, FL 32822

Phone: +1 407 812-OPEN (6736)

Fax: +1 407 517-4575

http://www.pentaho.com

Sales Inquiries: sales@pentaho.com

Introduction

Prerequisites

Pentaho Data Integration (PDI) can be configured to use your implementation of LDAP, MSAD, Apache DS, or Kerberos to authenticate users and authorize data access. You can also configure PDI to use a Single Sign On (SSO) framework or use a combination of these approaches.

Before you implement advanced security, you should have installed and configured the DI Server and Spoon, which is the PDI design tool.

You should have administrative-level knowledge of the security provider you want to use, details about your user community, and a plan for the user roles to be used in PDI. You should also know how to use the command line to issue commands for Microsoft Windows or Linux.

You will need a text editor to modify text files. You might also need to work on the actual machine that has DI software installed.

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

Table 1. Security Decision Table

Explore Considerations	Choose Options	
	Pentaho Security	Advanced Security Providers—LDAP, Single Sign-On, or Microsoft Active Directory
Summary	Pentaho Security is the easiest way to configure security quickly. Spoon enables you to define and manage users and roles. The DI Server controls which users and roles can access resources in the DI repository. Pentaho Security works well if you do not have a security provider or if you have a user community with less than 100 users.	If you are already using a security provider, such as LDAP, Single Sign-On, or Microsoft Active Directory, you can use the users and roles you have already defined with Pentaho. Your security provider controls which users and roles can access the DI repository. Advanced security scales well for production and enterprise user communities.
Expertise	Knowledge of your user community and which users should have which	Knowledge of your user community and which users should have which

Explore Considerations	Choose Options	
	Pentaho Security	Advanced Security Providers—LDAP, Single Sign-On, or Microsoft Active Directory
	roles in the Pentaho system. Knowledge about security in general is *not* required.	roles in the Pentaho system. Knowledge about your particular security provider and its options is required.
Time	It takes approximately 5 minutes per user and role to configure Pentaho Security.	It takes approximately 1 hour to configure the DI Server to use your existing security provider.
Recommendation	**Recommended for the Pentaho Trial Download, evaluating, and rapid development.**	**Recommended for production.**

Related Articles

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

- Administer DI Server
- Troubleshoot DI Server Issues

Configure LDAP for the DI Server

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

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

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

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

```
contextSource.password=secret
```

4. Update **adminRole** and **adminUser** for your system, replacing **adminRole** with the administrator role that you have
defined in your LDAP server, and replacing **adminUser** with the user name that has the administrator role assigned
to it.

```
adminRole=cn\=Administrator,ou\=roles
adminUser=uid\=admin,ou\=users
```

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

The DI Server is now configured to authenticate users against your directory server.
The LDAP Properties reference article contains supplemental information for LDAP values.

LDAP Properties

You can manually configure LDAP values by editing the `/pentaho-solutions/system/applicationContext-security-ldap.properties` file in the DI Server directory.

Connection Information (Context)

These entries define the connection to the LDAP server and the user/password used to perform directory searches against it.

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

Users

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

The `{0}` token is replaced by the username from the login dialog.

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 is appended automatically to the defined value here.

LDAP Property	Purpose	Example
userSearch.searchBase	Base (by username) for user searches	userSearch.searchBase=CN=Users
userSearch.searchFilter	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	userSearch.searchFilter=(sAMAccountName={0})

	usernames are represented 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.

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:1.2.840.113556.1.4.1941:=({0}))
populator.rolePrefix	A prefix to add to the beginning of the role name found in the group role attribute; the value can be an empty string.	populator.rolePrefix=
populator.searchSubtree	Indicates whether or not the search must include the current object and all children. If set to false, the search must include the current object only.	populator.searchSubtree=true

All Authorites Search

These entries populate roles that appear in the **Admin** tab . 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=group)

All User Name Search

These entries populate the users that appear on the **Admin** tab and can *only* be set manually in the `/pentaho-solutions/system/applicationContext-security-ldap.properties` file. These entities are not made available in the User Console.

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

Manual JDBC Connection Configuration

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

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

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

1. Stop the BA Server by running the **stop-pentaho** script.
2. Open `/pentaho-solutions/system/security.properties` with a text editor.
3. Change the value of the provide property to `jdbc`.
4. Set up the connection to the database that holds the user/authorities.
   a. Open the `/pentaho-solutions/system/applicationContext-spring-security-jdbc.properties` file with a text editor. Find these two lines and change the **jdbcDriver** and **URL** as appropriate.
   ```
   datasource.driver.classname=org.hsqldb.jdbcDriver
   ```
   ```
   datasource.url=jdbc:hsqldb:hsql://localhost:9002/userdb
   ```
   b. Change the user name and password by editing these two items.
   ```
   \datasource.username=sa, datasource.password=
   ```
   c. Set the **validation query** by editing this row. There are examples of different validation queries in the file.
   ```
   datasource.validation.query=SELECT 1 FROM INFORMATION_SCHEMA.SYSTEM_
   USERS
   ```
   d. Set the **wait timeout**, **max pool**, and **max idle** by editing these three items to change the defaults.
   ```
   datasource.pool.max.wait=-1, datasource.pool.max.active=8, datasource.
   max.idle=4
   ```
   e. Save the file and close the editor.
5. If you need to, modify these two queries that pull information about users/authorities.
   a. Open `/pentaho-solutions/system/applicationContext-spring-security-jdbc.xml` with a text editor.
   b. Find this line and change the **query** that returns the user and roles that the user is a member of as appropriate.

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

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

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

6. If you need to, modify these three queries that pull information about users/authorities.
    a. Open the `/pentaho-solutions/system/applicationContext-pentaho-security-jdbc.xml` file with a text editor.
    b. Find this line and change the **query** that shows the roles for security on objects as appropriate.

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

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

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

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

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

    e. Save the file and close the editor.
7. Update the default Pentaho admin user on the system to map to your JDBC admin user.
    a. Open the `/pentaho-solutions/system/repository.spring.properties` file with a text editor.
    b. Find these lines and change the default value from <admin> to map to your <admin username> in your JDBC system.

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

    c. Save the file and close the editor.

8. To fully map the JDBC's admin role to other configuration files, specify the name of the administrator role for your JDBC authentication database in the `applicationContext-pentaho-security-jdbc.xml` file.
    a. Open the `/pentaho-solutions/system/applicationContext-pentaho-security-jdbc.xml` file with a text editor.
    b. Find these lines and change the entry key to the key assigned to the administrator role in your JDBC authentication database.

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

    c. Save and close the file.
9. Start the server by running the **start-pentaho** script.

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

Create LDAP/JDBC Hybrid Configuration for the DI Server

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 DI 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 DI Server and Spoon.
2. Open `/pentaho-solutions/system/security.properties` with a text editor, then change the value of the property provider to `ldap`.
3. Open the `/pentaho-solutions/system/pentahoObjects.spring.xml` with a text editor, then find this code block and change the providerName to `jdbc`.

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

   <pen:publish as-type="INTERFACES">
     <pen:attributes>
    <pen:attr key="priority" value="50"/>
      </pen:attributes>
    </pen:publish>

    </pen:bean>
```

4. Edit the `/pentaho-solutions/system/applicationContext-pentaho-security-jdbc.xml` file and add the following two bean definitions, changing the connection and JDBC details to match your security database.

```
<bean id="dataSource"
class="org.springframework.jdbc.datasource.DriverManagerDataSource">
```

```
     <property name="driverClassName" value="org.
hsqldb:hsql://localhost:9002/userdb" />
     <property name="url" value="jdbc:hsqldb:hsql://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>
```

5. Save and close the file, then open `/pentaho-solutions/system/applicationContext-pentaho-security-jdbc.xml`. Find this code block and change `Admin` to an appropriate administrator role in your JDBC authentication database.

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

6. Close `applicationContext-pentaho-security-jdbc.xml`.
7. Open `/pentaho-solutions/system/applicationContext-springsecurity-ldap.xml` file and replace the populator bean definition with this one.

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

8. Delete the `/tomcat/work/` and `/tomcat/temp/` directories.
9. If needed, configure the Pentaho LDAP connection as explained in LDAP Properties.

10. Start the DI Server and Spoon, then log into Spoon.

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

Configure Microsoft Active Directory for the DI Server

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

Binding

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

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

Kerberos notation example for pentahoadmin@mycompany.com:

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

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

Windows domain notation example for MYCOMPANY\pentahoadmin:

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

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

Referrals

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

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

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

User DN Patterns vs. User Searches

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

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

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

User DN Pattern example:

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

In user searches, the **sAMAccountName** attribute should be used as the 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>
```

Nested Groups

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

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

This will search down the whole tree of nested groups.

Implement Kerberos Authentication

If your Hadoop clusters or MongoDB installation is secured using Kerberos, you can configure the Spoon and DI Server nodes so that users can access them.

- Use Kerberos Authentication to Provide Spoon Users Access to Hadoop Cluster
- Use Kerberos Authentication to Provide Spoon Users Access to MongoDB

Use Kerberos Authentication to Provide Spoon Users Access to Hadoop Cluster

If you use Kerberos to authenticate access to your Hadoop cluster, with a little extra configuration, you can also use Kerberos to authenticate Spoon users who attempt the access the cluster through a step in the transformation. When a user attempts to run a transformation that contains a step that connects to a Hadoop cluster to perform a function, the user's account credential is matched against the credentials in the Kerberos administrative database on the Hadoop cluster. If the credentials match, the Kerberos Key Distribution Center (KDC) grants an authorization ticket and access is granted. If not, the user is not authentication and the step does not run.

To set up Kerberos authentication to provide Spoon users with access to the Hadoop cluster, you will need to perform four sets of tasks.

- Complete Cluster and Client-Node Prerequisites
- Add Users To Kerberos Database on Hadoop Cluster
- Set Up Kerberos Administrative Server and KDC to Start When the Server Starts
- Configure your Hadoop Cluster
- Configure Spoon Client Side Nodes
- Test Authentication from within Spoon

Complete Cluster and Client-Node Prerequisites

Make sure that you have completed the following tasks before you move to the next section.

- Install a Hadoop cluster on one or more Linux servers. The cluster should be running one of the versions of Hadoop listed in the Configuring Pentaho for your Hadoop Distro and Version section of the Pentaho Big Data wiki.
- Configure the Hadoop cluster with a Kerberos Realm, Kerberos KDC, and Kerberos Administrative Server.
- Make sure the Hadoop cluster, including the name node, data nodes, secondary name node, job tracker, and task tracker nodes have been configured to accept remote connection requests.
- Make sure the Kerberos clients have been set up for all data, task tracker, name, and job tracker nodes if you are have deployed Hadoop using an enterprise-level program.
- Install the current version of Spoon on each client machine.
- Make sure each client machine can use a hostname to access the Hadoop cluster. You should also test to ensure that IP addresses resolve to hostnames using both forward and reverse lookups.

Add Users to Kerberos Database on Hadoop Cluster

Add the user account credential for each Spoon user that should have access to the Hadoop cluster to the Kerberos database. You only need to do this once.

1. Log in as root (or a privileged user), to the server that hosts the Kerberos database.
2. Make sure there is an operating system user account on *each node* in the Hadoop cluster for *each user* that you want to add to the Kerberos database. Add operating system user accounts if necessary. Note that the user account UIDs *must be greater* than the minimum user ID value (`min.user.id`). Usually, the minimum user ID value is set to 1000.
3. Add user identification to the Kerberos database by completing these steps.
    a. Open a **Terminal** window, then add the account username to the Kerberos database, like this. The name should match the operating system user account that you verified (or added) in the previous step. If successful, a message appears indicating that the user has been created.

    ```
    root@kdc1:~# kadmin.local -q "addprinc <username>"
    ...
    Principal "<user name>@DEV.LOCAL" created.
    ```

    b. Repeat for each user you want to add to the database.

Set Up Kerberos Administrative Server and KDC to Start When Server Starts

It is a good practice to start the Kerberos Administrative Server and the KDC when the server boots. One way to do this is to set them up to run as a service. This is an optional, but recommended step.

1. If you have not done so already, log into the server that contains the Kerberos Administrative Server and the KDC.
2. Set the Kerberos Administrative Server to run as a service when the system starts. By default, the name of the Kerberos Administrative Server is `kadmin`. If you do not know how to do this, check the documentation for your operating system.
3. Set the KDC to run as a service when the system starts. By default, the name of the KDC is `krb5kdc`.

Make HDP-Specific Configurations

If you are using HDP, you might require additional configuration changes depending on your environment. Before you complete the steps in this section, review the installation and configuration documentation at the HDP documentation website: http://docs.hortonworks.com/.

NOTE:

The instructions in this section are for configuring a test server only. Adjust the following instructions to meet the needs of your HDP test and production environments.

HDP-specific configuration changes are divided into two groups.

- Edit HDP Cluster-Side Configuration Files
- Set Access Permissions

Edit HDP Cluster-Side Configuration Files

Three files should be edited on the cluster: core-site.xml, hdfs-site.xml, and oozie.xml.

Edit core-site.xml

Edit the core-site.xml file to specify the appropriate hosts and groups for various proxy users. To edit the file do the

1. On the cluster, open the `core-site.xml` file in a text editor. By default, it is in the `$HADOOP_CONF_DIR`.
2. Set the host and group for proxyusers using the following example as a guide.

- `hadoop.proxyuser.{username of proxyuser such as hive or oozie)}.hosts=FQHN of the hadoop manager node (by default)`
- `hadoop.proxyuser.{username of proxyuser such as hive or oozie)}.groups=users(by default)`

Here is an example. Modify the `samplehost`, `samplecompany`, and `samplegroup` values to match your environment. Note that not all of the properties might be needed. Also, note that the kinit_user should already be added to any groups that you specify.

```
<property>
  <name>hadoop.proxyuser.root.hosts</name>
  <value>samplehost.samplecompany.com</value>
</property>
<property>
  <name>hadoop.proxyuser.falcon.hosts</name>
  <value>samplehost.samplecompany.com</value>
</property>
<property>
   <name>hadoop.proxyuser.hive.hosts</name>
   <value>samplehost.samplecompany.com</value>
</property>
<property>
   <name>hadoop.proxyuser.HTTP.hosts</name>
   <value>samplehost.samplecompany.com</value>
</property>
<property>
   <name>hadoop.proxyuser.oozie.hosts</name>
   <value>samplehost.samplecompany.com</value>
</property>
<property>
   <name>hadoop.proxyuser.hcat.hosts</name>
   <value>samplehost.samplecompany.com</value>
</property>
<property>
   <name>hadoop.proxyuser.root.groups</name>
```

```
   <value>samplegroup</value>
</property>
<property>
   <name>hadoop.proxyuser.oozie.groups</name>
   <value>samplegroup</value>
</property>
<property>
   <name>hadoop.proxyuser.HTTP.groups</name>
   <value>samplegroup</value>
</property>
<property>
   <name>hadoop.proxyuser.falcon.groups</name>
   <value>samplegroup</value>
</property>
<property>
   <name>hadoop.proxyuser.hcat.groups</name>
   <value>samplegroup</value>
</property>
<property>
   <name>hadoop.proxyuser.hive.groups</name>
   <value>samplegroup</value>
</property>
```

0. 1. Save and close the file.

Edit hdfs-site.xml

To edit the `hdfs-site.xml` file, complete these steps.

1. On the cluster, open the `hdfs-site.xml` file with a text editor. By default it is in the `$HADOOP_CONF_DIR`.
2. Set the `dfs.nfs.exports.allowed.hosts` property's value to read and write, like this (note the space between the asterisk "`*`" and the "`rw`" in the following code snippet):

```
<property>
      <name>dfs.nfs.exports.allowed.hosts</name>
      <value>* rw</value>
    </property>
```

0. 1. Save and close the file.

Edit oozie-site.xml

To edit the `oozie-site.xml` file, complete these steps.

1. On the cluster, open the `oozie-site.xml` file in a text editor. By default, it is located in the `$OOZIE_CONF_DIR`.
2. Set the hosts and groups for proxyusers using the following example as a guide.

```
<property>
<name>hadoop.proxyuser.oozie.hosts</name>
<value>samplehost.samplecompany.com</value>
</property>
<property>
<name>hadoop.proxyuser.oozie.group</name>
<value>samplegroup</value>
</property>
<property>
<name>hadoop.proxyuser.{kinit_user}.hosts</name>
<value>samplehost.samplecompany.com</value>
</property>
<property>
<name>hadoop.proxyuser.{kinit_user}.group</name>
<value>samplegroup</value>
</property>
<property>
oozie.service.ProxyUserService.proxyuser.{kinit_user}.hosts</name>
      <value>samplehost.samplecompany.com</value>
    </property>
<property>
oozie.service.ProxyUserService.proxyuser.{kinit_user}.group</name>
      <value>samplegroup</value>
    </property>
```

2. 1. Save, then close the file.

Set Access Permissions

Access permissions must be granted to allow the knit_user access to the appropriate file systems. The examples in this section are specific to test servers only. Increase security by modifying the instructions and parameters are required for your organization.

Grant kinit_user Access to Hadoop File System

Ensure that the kinit_user has permission to access the hadoop file system and any other directories where access is required.

In a terminal window or shell tool, enter a chmod command to grant the kinit_user access to the appropriate directories, like this:

```
hadoop fs -chmod -R 777 /user/{kinit_user}
```

NOTE:

To make this more secure, use a different value than 777.

Edit YARN ACL, Logs, and Timeline Server Settings

To edit this settings, complete these steps.

1. In the YARN resource manager, set the yarn.acle.enable to either true or false, as needed.
2. Set the yarn.admin.acl to *
3. Set yarn-log-aggregation-enable to false.
4. Set yarn.timeline-service.enabled to false.
5. Save and close the resource manager file.

Grant Privileges to Hive Mysql database

In a shell or terminal window on the Hadoop cluster, enter the following.

```
mysql
grant all privileges on *.* to 'hive'@'ip of the kettle client host' with grant
option; //this should be done for all client hosts or could be specified '%' to
allow all this connection from all hosts
exit;
hive
GRANT ALL ON DATABASE default TO USER {kinit_user} WITH GRANT OPTION; //instead
of 'default' could be other databases
```

Edit the Hive-Site.xml to Grant Owner Authorization For Tables

To edit the hive-site.xml file, do these things.

1. Open the hive-site.xml file in a text editor.
2. Set the hive.security.authorization.createable.owner.grants property to ALL.
3. Save and close the file.

Grant HBASE privileges

To grant HBASE privileges, in a shell or terminal window, type the following:

```
hbase shell
grant '{kinit_user}','RWXCA'
```

Configure Spoon Client-Side Nodes

After you have added users to the database and configured the Kerberos admin and KDC to start when the server starts, you are ready to configure each client-side node from which a user might access the Hadoop cluster. Client-side nodes should each have a copy of Spoon already installed. Client-side configuration differs based on your operating system.

- Configure Linux and Mac Client Nodes
- Configure Windows Client Nodes

Configure Linux and Mac Client Nodes

To configure Linux client nodes, complete these tasks.

- Install JCE on Linux and Mac Clients
- Configure PDI for Hadoop Distribution and Version on Linux and Mac Clients
- Download and Install Kerberos on Linux and Mac Clients
- Modify Kerberos Configuration File to Reflect Realm, KDC, and Admin Server
- Synchronize Clock on Linux and Mac Clients
- Obtain Kerberos Ticket on Linux and Mac Clients

Install JCE on Linux and Mac Clients

This step is optional. The KDC configuration includes an AES-256 encryption setting. If you want to use this encryption strength, you will need to install the Java Cryptographic Extension (JCE) files.

1. Download the Java Cryptographic Extension (JCE) for the currently supported version of Java from the Oracle site.
2. Read the installation instructions that are included with the download.
3. Copy the JCE jars to the `java/lib/security` directory where PDI is installed on the Linux client machine.

Configure PDI for Hadoop Distribution and Version on Linux and Mac Clients

To configure DI to connect to the Hadoop cluster, you'll need to copy Hadoop configuration files from the cluster's **name** node to the appropriate place in the `hadoop-configurations` subdirectory.

1. Back up the `core-site.xml`, `hdfs-site.xml`, and `mapred-site.xml` files that are in the `design-tools/data-integration/plugins/pentaho-big-data-plugin/hadoop-configurations/<directory of the shim that is in your plugin.properties file>`.
2. Copy the `core-site.xml`, `hdfs-site.xml`, and `mapred-site.xml` from the cluster's name node to this directory on each client: `design-tools/data-integration/plugins/pentaho-big-data-plugin/hadoop-configurations/<directory of the shim that is in your plugin.properties file>`. **Note:** If you made configuration changes to the core-site.xml, hdfs-site.xml, or mapred-site.xml files previously, you will need to make those changes again. Reference your backed up copies of the files if needed.

Download and Install Kerberos Client on Linux and Mac Clients

Download and install a Kerberos client. Check your operating system's documentation for further details on how to do this.

Modify Kerberos Configuration File to Reflect Realm, KDC, and Admin Server on Linux and Mac Clients

Modify the Kerberos configuration file to reflect your Realm, KDC, and Admin Server.

1. Open the `krb5.conf` file. By default this file is located in `/etc/krb5.conf`, but it might appear somewhere else on your system.
2. Add your Realm, KDC, and Admin Server information. The information in-between the carats **< >** indicates where you should modify the code to match your specific environment.

```
[libdefaults]
       default_realm = <correct default realm name>
clockskew = 300
v4_instance_resolve = false
v4_name_convert = {
host = {
rcmd = host
ftp = ftp
}
plain = {
something = something-else
}
}

[realms]
<correct default realm name>= {
kdc=<KDC IP Address, or resolvable Hostname>
admin_server=< Admin Server IP Address, or resolvable Hostname>
}
MY.REALM = {
kdc = MY.COMPUTER
}
OTHER.REALM = {
v4_instance_convert = {
kerberos = kerberos
computer = computer.some.other.domain
}
}
[domain_realm]
.my.domain = MY.REALM
```

3. Save and close the configuration file.
4. Restart the computer.

Synchronize Clock on Linux and Mac Clients

Synchronize the clock on the Linux client with the clock on the Hadoop cluster. This is important because if the clocks are too far apart, then when authentication is attempted, Kerberos will not consider the tickets that are granted to be valid and the user will not be authenticated. The times on the Linux client clock and the Hadoop cluster clock must not be greater than the range you entered for the `clockskew` variable in `krb5.conf` file when you completed the steps in the Modify Kerberos Configuration File to Reflect Realm, KDC, and Admin Server on Linux Client task.

Consult your operating system's documentation for information on how to properly set your clock.

Obtain Kerberos Ticket on Linux and Mac Clients

To obtain a Kerberos ticket, complete these steps.

1. Open a **Terminal** window and type `kinit` at the prompt.
2. When prompted for a password, enter it.
3. The prompt appears again. To ensure that the Kerberos ticket was granted, type `klist` at the prompt.
4. Authentication information appears.

Configure Windows Client Nodes

To configure Windows client nodes, complete these tasks.

- Install JCE on Windows Client
- Configure PDI for Hadoop Distribution and Version on Windows Client
- Download and Install Kerberos on Windows Client
- Modify Kerberos Configuration File to Reflect Realm, KDC, and Admin Server
- Synchronize Clock on Windows Client
- Obtain Kerberos Ticket on Windows Client

Install JCE on Windows Client

This step is optional. The KDC configuration includes an AES-256 encryption setting. If you want to use this encryption strength, you will need to install the Java Cryptographic Extension (JCE) files.

1. Download the Java Cryptographic Extension (JCE) for the currently supported version of Java from the Oracle site.
2. Read the installation instructions that are included with the download.
3. Copy the JCE jars to the `java\lib\security` directory where PDI is installed.

Configure PDI for Hadoop Distribution and Version on Windows Client

To configure PDI to connect to the Hadoop cluster, you'll need to copy Hadoop configuration files from the cluster's **name** node to the appropriate place in the `hadoop-configurations` subdirectory.

1. Back up the `core-site.xml`, `hdfs-site.xml`, and `mapred-site.xml` files that are in the `design-tools/data-integration/plugins/pentaho-big-data-plugin/hadoop-configurations/<directory of the shim that is in your plugin.properties file>`.
2. Copy the `core-site.xml`, `hdfs-site.xml`, and `mapred-site.xml` from the cluster's name node to this directory on each client: `design-tools/data-integration/plugins/pentaho-big-data-plugin/hadoop-configurations/<directory of the shim that is in your plugin.properties file>`. **Note:** If you made configuration changes to the core-site.xml, hdfs-site.xml, or mapred-site.xml files previously, you will need to make those changes again. Reference your backed up copies of the files if necessary.

Download and Install Kerberos Client on Windows Client

Download and install a Kerberos client. We recommend that you use the Heimdal implementation of Kerberos, which can be found here: https://www.secure-endpoints.com/heimdal/.

Modify Kerberos Configuration File to Reflect Realm, KDC, and Admin Server on Windows Client

You will need to modify the Kerberos configuration file to reflect the appropriate realm, KDC, and Admin Server.

1. Open the krb5.conf file. By default this file is located in `c:\Program Data\Kerberos`. This location might be different on your system.
2. Add the appropriate realm, KDC, and Admin Server information. An example of where to add the data appears below.

```
[libdefaults]
        default_realm = <correct default realm name>
clockskew = 300
v4_instance_resolve = false
v4_name_convert = {
host = {
rcmd = host
ftp = ftp
}
plain = {
something = something-else
}
}

[realms]
<correct default realm name>= {
kdc=<KDC IP Address, or resolvable Hostname>
admin_server=< Admin Server IP Address, or resolvable Hostname>
}
MY.REALM = {
kdc = MY.COMPUTER
}
OTHER.REALM = {
v4_instance_convert = {
kerberos = kerberos
computer = computer.some.other.domain
}
}
[domain_realm]
.my.domain = MY.REALM
```

3. Save and close the configuration file.
4. Make a copy of the configuration file and place it in the `c:\Windows` directory. Rename the file `krb5.ini`.
5. Restart the computer.

Synchronize Clock on Windows Client

Synchronize the clock on the Windows client with the clock on the Hadoop cluster. This is important because if the clocks are too far apart, then when authentication is attempted, Kerberos will not consider the tickets that are granted to be valid and the user will not be authenticated. The times on the Windows client clock and the Hadoop cluster clock must not be greater than the range you entered for the `clockskew` variable in `krb5.conf` file when you completed the steps in the Modify Kerberos Configuration File to Reflect Realm, KDC, and Admin Server on Windows Client task.

Consult your operating system's documentation for information on how to properly set your clock.

Obtain Kerberos Ticket on Windows Client

To obtain a Kerberos ticket, complete these steps.

1. Open a **Command Prompt** window and type `kinit` at the prompt.
2. When prompted for a password, enter it.
3. The prompt appears again. To ensure that the Kerberos ticket was granted, type `klist` at the prompt.
4. Authentication information appears.

Test Authentication from Within Spoon

To test the authentication from within Spoon, run a transformation that contains a step that connects to a Hadoop cluster. For these instructions to work properly, you should have read and write access to the your home directory on the Hadoop cluster.

1. Start Spoon.
2. Open an existing transformation that contains a step to connect to the Hadoop cluster. If you don't have one, consider creating something like this.
   a. Create a new transformation.
   b. Drag the **Generate Rows** step to the canvas, open the step, indicate a limit (the number of rows you want to generate), then put in field information, such as the name of the field, type, and a value.
   c. Click **Preview** to ensure that data generates, then click the **Close** button to save the step.
   d. Drag a **Hadoop File Output** step onto the canvas, then draw a hop between the **Generate Rows** and **Hadoop File Output** steps.
   e. In the **Filename** field, indicate the path to the file that will contain the output of the **Generate Rows** step. The path should be on the Hadoop cluster. Make sure that you indicate an extension such as `txt` and that you want to create a parent directory and that you want to add filenames to the result.
   f. Click the **OK** button then save the transformation.
3. Run the transformation. If there are errors correct them.

4. When complete, open a **Terminal** window and view the results of the output file on the Hadoop filesystem. For example, if you saved your file to a file named `test.txt`, you could type a command like this:

```
hadoop fs -cat /user/pentaho-user/test/test.txt
```

Use Kerberos Authentication to Provide Spoon Users Access to MongoDB

If you use Kerberos to authenticate access to your installation of MongoDB, with a little extra configuration, you can also use Kerberos to authenticate Spoon users who attempt the access MongoDB through a step in a transformation. When a user attempts to run a transformation that contains a step that connects to a MongoDB cluster to perform a function, the credentials in the step are matched against the credentials in the Kerberos administrative database on MongoDB. If the credentials match, the Kerberos Key Distribution Center (KDC) grants an authorization ticket and access is granted. If not, the user is not authentication and the step does not run.

To set up Kerberos authentication to provide Spoon users with access to MongoDB you will need to perform several sets of tasks.

- Complete MongoDB and Client Prerequisites
- Add Users To Kerberos Database
- Set Up Kerberos Administrative Server and KDC to Start When the Server Starts
- Configure Spoon Client Side Nodes
- Test Authentication from within Spoon

Complete MongoDB and Client Prerequisites

Make sure that you have completed the following tasks before you move to the next section.

- Make sure that you have installed and configured an *Enterprise* version MongoDB according to the instructions in the MongoDB installation guide http://docs.mongodb.org/manual/installation/.
- Configure MongoDB to use Kerberos. Instructions for how to do that appear here: http://docs.mongodb.org/manual/tutorial/control-access-to-mongodb-with-kerberos-authentication/.
- Install the current version of Spoon on each client machine.
- Make sure each client machine can use a hostname to access MongoDB. You should also test to ensure that IP addresses resolve to hostnames using both forward and reverse lookups.

Add Users to Kerberos Database

Add the user account credential to the Kerberos database for each Spoon user that should have access to MongoDB. You only need to do this once for each user.

1. Log in as root (or a privileged user), to the server that hosts the Kerberos database.
2. Add user identification to the Kerberos database by completing these steps.
    a. Open a **Terminal** window.

b. Add the account username to the Kerberos database, like this. The username should match the one used to create the user in MongoDB. See the create users part of http://docs.mongodb.org/manual/tutorial/control-access-to-mongodb-with-kerberos-authentication/ for more details. If successful, a message appears indicating that the user has been created.

```
root@kdc1:~# kadmin.local -q "addprinc <username>"
...
Principal "<user name>@DEV.LOCAL" created.
```

c. Repeat for each user you want to add to the database.

Set Up Kerberos Administrative Server and KDC to Start When Server Starts

It is a good practice to start the Kerberos Administrative Server and the KDC when the server boots. One way to do this is to set them up to run as a service. This is an optional, but recommended step.

1. If you have not done so already, log into the server that contains the Kerberos Administrative Server and the KDC.
2. Set the Kerberos Administrative Server to run as a service when the system starts. By default, the name of the Kerberos Administrative Server is `kadmin`. If you do not know how to do this, check the documentation for your operating system.
3. Set the KDC to run as a service when the system starts. By default, the name of the KDC is `krb5kdc`.

Configure Spoon Client-Side Nodes

After you have added users to the database and configured the Kerberos admin and KDC to start when the server starts, you are ready to configure each client-side node from which a user might access MongoDB. Client-side nodes should each have a copy of Spoon already installed. Client-side configuration differs based on your operating system.

- Configure Linux Client Nodes
- Configure Windows Client Nodes

Configure Linux and Mac Client Nodes

To configure Linux client nodes, complete these tasks.

- Install JCE on Linux and Mac Clients
- Download and Install Kerberos on Linux and Mac Clients
- Modify Kerberos Configuration File to Reflect Realm, KDC, and Admin Server
- Specify the Location of the Kerberos Configuration File on Mac Clients that Run Spoon
- Specify the Location of the Kerberos Configuration File on Mac Clients that Run PRD
- Synchronize Clock on Linux and Mac Clients
- Obtain Kerberos Ticket on Linux and Mac Clients

Install JCE on Linux and Mac Clients

This step is optional. The KDC configuration includes an AES-256 encryption setting. If you want to use this encryption strength, you will need to install the Java Cryptographic Extension (JCE) files.

1. Download the Java Cryptographic Extension (JCE) for the currently supported version of Java from the Oracle site.
2. Read the installation instructions that are included with the download.
3. Copy the JCE jars to the `java/lib/security` directory where PDI is installed on the Linux client machine.

Download and Install Kerberos Client on Linux and Mac Clients

Download and install a Kerberos client. Check your operating system's documentation for further details on how to do this.

Modify Kerberos Configuration File to Reflect Realm, KDC, and Admin Server on Linux and Mac Clients

Modify the Kerberos configuration file to reflect your Realm, KDC, and Admin Server.

1. Open the `krb5.conf` file. By default this file is located in `/etc/krb5.conf`, but it might appear somewhere else on your system.
2. Add your Realm, KDC, and Admin Server information. The information in-between the carats < > indicates where you should modify the code to match your specific environment.

```
[libdefaults]
       default_realm = <correct default realm name>
clockskew = 300
v4_instance_resolve = false
v4_name_convert = {
host = {
rcmd = host
ftp = ftp
}
plain = {
something = something-else
}
}

[realms]
<correct default realm name>= {
kdc=<KDC IP Address, or resolvable Hostname>
admin_server=< Admin Server IP Address, or resolvable Hostname>
}
MY.REALM = {
kdc = MY.COMPUTER
}
OTHER.REALM = {
v4_instance_convert = {
kerberos = kerberos
```

```
computer = computer.some.other.domain
}
}
[domain_realm]
.my.domain = MY.REALM
```

3. Save and close the configuration file.
4. Restart the computer.

Specify the Location of the Kerberos Configuration File on Mac Clients That Run Spoon

If you are configuring Spoon to use Kerberos to authenticate MongoDB on a Mac client, you might need to manually specify where the Kerberos configuration file can be found. Do this if the version of the JRE that the Spoon uses is earlier than Java 1.70_40, because the JRE attempts to find the Kerberos Configuration file in a different location than the default.

1. Use **Finder** to navigate to `design-tools/data-integration/launcher/launcher.properties` file.
2. In the `launcher.properties` file, add a java parameter that indicates the realm and the KDC that you specified in the [Modify Kerberos Configuration File to Reflect Realm, KDC, and Admin Server](#) step. Make sure to set both of these properties.

```
-Djava.security.krb5.realm=<Kerberos Realm>
-Djava.secrutiy.krb5.kdc=&lt;Kerberos KDC>
```

3. If you need to set additional configuration properties for your Kerberos installation, see Locating the krb5.conf Configuration File section located in http://docs.oracle.com/javase/7/docs/technotes/guides/security/jgss/tutorials/KerberosReq.html for details.
4. Close and save the `launcher.properties` file.

Specify the Location of the Kerberos Configuration File on Mac Clients that Run PRD

If you are configuring the PRD to use Kerberos to authenticate MongoDB on a Mac, you will need to manually specify where the Kerberos Configuration file can be found. You must do this if the version of the JRE that the PRD uses is earlier than Java 1.7.0_40, because it attempts to find the Kerberos Configuration file in a different location than the default.

1. Use **Finder** to navigate to the `Pentaho Report Designer.app` file which is in the `design-tools` directory. Right click and select **Show Package Contents**.
2. Navigate to the **Contents > Java**.
3. Open `launcher.properties`. Do not use the `launcher.properties` file that is in the root of the `app` directory.
4. In the `launcher.properties` file, add a java parameter that indicates the realm and the KDC that you specified in the [Modify Kerberos Configuration File to Reflect Realm, KDC, and Admin Server](#) step. Make sure to set both of these properties.

```
-Djava.security.krb5.realm=<Kerberos Realm>
-Djava.secrutiy.krb5.kdc=&lt;Kerberos KDC>
```

5. If you need to set additional configuration properties for your Kerberos installation, see Locating the krb5.conf Configuration File section located in http://docs.oracle.com/javase/7/docs/technotes/guides/security/jgss/tutorials/KerberosReq.html for details.
6. Close and save the `launcher.properties` file.

Synchronize Clock on Linux Client

Synchronize the clock on the Linux client with the clock on MongoDB host. This is important because if the clocks are too far apart, then when authentication is attempted, Kerberos will not consider the tickets that are granted to be valid and the user will not be authenticated. The times on the Linux client clock and the MongoDB host clock must not be greater than the range you entered for the `clockskew` variable in `krb5.conf` file when you completed the steps in the Modify Kerberos Configuration File to Reflect Realm, KDC, and Admin Server on Linux Client task.

Consult your operating system's documentation for information on how to properly set your clock.

Obtain Kerberos Ticket on Linux Client

To obtain a Kerberos ticket, complete these steps.

1. Open a **Terminal** window and type `kinit` at the prompt.
2. When prompted for a password, enter it.
3. The prompt appears again. To ensure that the Kerberos ticket was granted, type `klist` at the prompt.
4. Authentication information appears.

Configure Windows Client Nodes

To configure Windows client nodes, complete these tasks.

- Install JCE on Windows Client
- Download and Install Kerberos on Windows Client
- Modify Kerberos Configuration File to Reflect Realm, KDC, and Admin Server
- Synchronize Clock on Windows Client
- Obtain Kerberos Ticket on Windows Client

Install JCE on Windows Client

This step is optional. The KDC configuration includes an AES-256 encryption setting. If you want to use this encryption strength, you will need to install the Java Cryptographic Extension (JCE) files.

1. Download the Java Cryptographic Extension (JCE) for the currently supported version of Java from the Oracle site.
2. Read the installation instructions that are included with the download.
3. Copy the JCE jars to the `java\lib\security` directory where PDI is installed.

Download and Install Kerberos Client on Windows Client

Download and install a Kerberos client. We recommend that you use the Heimdal implementation of Kerberos, which can be found here: https://www.secure-endpoints.com/heimdal/.

Modify Kerberos Configuration File to Reflect Realm, KDC, and Admin Server on Windows Client

You will need to modify the Kerberos configuration file to reflect the appropriate realm, KDC, and Admin Server.

1. Open the krb5.conf file. By default this file is located in `c:\Program Data\Kerberos`. This location might be different on your system.
2. Add the appropriate realm, KDC, and Admin Server information. An example of where to add the data appears below.

```
[libdefaults]
       default_realm = <correct default realm name>
clockskew = 300
v4_instance_resolve = false
v4_name_convert = {
host = {
rcmd = host
ftp = ftp
}
plain = {
something = something-else
}
}

[realms]
<correct default realm name>= {
kdc=<KDC IP Address, or resolvable Hostname>
admin_server=< Admin Server IP Address, or resolvable Hostname>
}
MY.REALM = {
kdc = MY.COMPUTER
}
OTHER.REALM = {
v4_instance_convert = {
kerberos = kerberos
computer = computer.some.other.domain
}
}
[domain_realm]
.my.domain = MY.REALM
```

3. Save and close the configuration file.
4. Make a copy of the configuration file and place it in the `c:\Windows` directory. Rename the file `krb5.ini`.

5. Restart the computer.

Synchronize Clock on Windows Client

Synchronize the clock on the Windows client with the clock on the MongoDB host. This is important because if the clocks are too far apart, then when authentication is attempted, Kerberos will not consider the tickets that are granted to be valid and the user will not be authenticated. The times on the Windows client clock and the MongoDB host's clock must not be greater than the range you entered for the `clockskew` variable in `krb5.conf` file when you completed the steps in the Modify Kerberos Configuration File to Reflect Realm, KDC, and Admin Server on Windows Client task.

Consult your operating system's documentation for information on how to properly set your clock.

Obtain Kerberos Ticket on Windows Client

To obtain a Kerberos ticket, complete these steps.

1. Open a **Command Prompt** window and type `kinit` at the prompt.
2. When prompted for a password, enter it.
3. The prompt appears again. To ensure that the Kerberos ticket was granted, type `klist` at the prompt.
4. Authentication information appears.

Test Authentication from Within Spoon

To test the authentication from within Spoon, create a transformation that contains a MongoDB Input that connects to MongoDB. For these instructions to work properly, you should have permission to read a database and the corresponding collections on the instance of MongoDB you want to connect to.

1. Start Spoon.
2. Create a new transformation.
3. Drag the **MongoDB Input** step to the canvas and open the step.
4. Enter the host name of the MongoDB instance and port for MongoDB.
5. In the **username** field, indicate the Kerberos principal, using this format: `<primary>/<instance>@KERBEROS_REALM`. (Be sure to include the forward slash. Also note that the Kerberos Realm is case sensitive.) Check with your administrator if you do not know your Kerberos principal.
6. Leave the **password** field blank.
7. Click the **Authenticate using Kerberos** checkbox.
8. Click the **Input** options tab, then enter the name of a database on MongoDB to which you have read permissions.
9. Click the **Get Collections** button. You should be able to see the databases you have read accessss to, as well as the collections in the drop down lists.
10. Click the **Preview** button. If you see data, then you know that Kerberos is working properly.

Use Impersonation to Access a MapR Cluster

By default, the DI Server admin user executes transformations and jobs. But, if your transformation or job needs to run on a MapR cluster or access its resources, the DI Server admin might not have an account there or have the right permissions and accesses.

Using impersonation helps solve this issue. With impersonation, you indicate that a transformation should run using the permissions and accesses of a different Hadoop user. Impersonation leverages the Hadoop user's existing permissions and accesses to provide access to components that run on MapR clusters such as mapreduce, pig, oozie, sqoop, hive, or a directory on HDFS.

There are a couple of limitations.

- Only the MapR super user can impersonate other users.
- With Linux, you use *impersonation* to specify different proxy users for HDFS, Mapreduce, Pig, Sqoop, Oozie, and Pentaho Map Reduce components.
- With Windows you can only specify a single, *spoofed* user.

Instructions for impersonation or spoofing depend on your Spoon client's operating system.

- Prerequisites for Impersonation and Spoofing for Both Linux and Windows Nodes
- Set Up Impersonation on Linux Client Node
- Set Up Spoofing on Windows Client Node

Prerequisites for Impersonation and Spoofing

Prerequisites for impersonation and spoofing include setting up and configuring the MapR distribution, setting up user accounts, then storing authorization provider credentials and overriding the Kerberos ticket cache.

- Set Up MapR Nodes
- Make MapR the Active Hadoop Distribution
- Set Up User Accounts
- Store Authorization Provider Credentials
- Override Kerberos Ticket Cache
- Set Hive Database Connection Parameters

Set Up MapR Nodes

Set up a MapR cluster and apply MapR security. See MapR documentation for installation and security instructions.

Make MapR the Active Hadoop Distribution

Make MapR your active Hadoop distribution and configure it. See Set Active Hadoop Distribution [Set Active Hadoop Distribution] and Additional Configuration for MapR Shims [Additional configuration for MapR Shims] for more detail.

Set Up User Accounts

- Set up user accounts for MapR and client nodes.
- Set up accounts for the users you want to impersonate or spoof on each MapR node. The usernames, passwords, UID, and GID should be the same on each node.
- For Linux Spoon client and DI Server nodes, set up accounts for the users you want to impersonate The usernames, passwords, UID, and GID should be the same on each node. *You do not have to do this for Windows Spoon client or DI Server nodes*.
- On both Windows and Linux nodes, add the impersonated or spoofed users to additional groups, if needed. Do this if users require access to resources restricted to members of a group. Ensure the group names and GIDs are correct and are the same for each node.

Store Authorization Provider Credentials

Store authorization provider credentials so you do not have to retype usernames, passwords, or other credential information each time you need them for a transformation step or job entry.

1. On the DI Server, open the `config.properties` file in `server/data-integration-server/pentaho-solutions/system/kettle/plugins/pentaho-big-data-plugin/hadoop-configurations/[hadoop distribution]`.

NOTE:

[hadoop distribution] is the name of the Hadoop distribution, such as mapr31.

1. Set the Kerberos Principal property.

```
authentication.kerberos.principal=user@omnicorp.com
```

1. Decide whether to authenticate using a password or a keytab file.

- To authenticate with a password, set the authentication.kerberos.password property.

```
authentication.kerberos.password=userPassword
```

NOTE:

Use Kettle encryption to store the password more securely.

- To authenticate with a keytab file, set the authentication.kerberos.keytabLocation property to the keytab file path.

```
authentication.kerberos.keytabLocation=/home/Server14/Kerberos/username.keytab
```

NOTE:

If both the authentication.kerberos.password and authentication.kerberos.keytabLocation properties are set, the authentication.kerberos.password property takes precedence.

1. Assign an ID to the authentication credentials that you just specified (Kerberos Principal and password or keytab), by setting the authentication.kerberos.id property.

```
authentication.kerberos.id=mapr-kerberos
```

1. To use authentication credentials you just specified, set the authentication.superuser.provider to the authentication.kerberos.id.

```
authentication.superuser.provider=mapr-kerberos
```

1. Save and close the file.
2. Repeat this process on Spoon. The config.properties file is in `design-tools/data-integration/plugins/pentaho-big-data-plugin/hadoop-configurations/[hadoop distribution]`.

Override Kerberos Ticket Cache

If you are logged into the Spoon host machine and your account has already been authenticated using Kerberos, indicate that you want to use the authentication information that is in the config.properties file instead, not the one that has already been saved in the Kerberos ticket cache.

1. Open the `mapr.login.conf` file on the host. By default, the file is located in `opt/mapr/conf`.
2. In the `hadoop_hybrid` section, set `useTicketCache` and `renewTGT` variables to false, like this:

```
hadoop_hybrid{
org.apache.hadoop.security.login.KerberosBugWorkAroundLoginModule optional
      useTicketCache=false
      renewTGT=false
```

1. Save and close the `mapr.login.conf` file.

Set Hive Database Connection Parameters

To access Hive, you need to set several database connection parameters from within Spoon.

1. Open the `hive-site.xml` file that is on the hive server host. Note the values for the `kerberos.principal` and the `sasl.qop`.
2. Close the `hive-site.xml` file.
3. Start Spoon.
4. In Spoon, open the **Database Connection** window.
5. Click Options. Add the following parameters and set them to the values that you noted in the `hive-site.xml` file.

- `sasl.qop`
- `principal`

NOTE:

The principal typically has a mapr prefix before the name, like this: `mapr/mapr31.pentaho@mydomain`

1. Click **OK** to close the window.

Set Up Impersonation on Linux Client Node

To set up impersonation on a Linux client node, specify proxy users in the core-site.xml file.

1. On the DI Server node, open the `core-site.xml` file in `pentaho-solutions/system/kettle/plugins/pentaho-big-data-plugin/hadoop-configurations/[hadoop distribution]`.

NOTE:

[hadoop configuration] is the name of the Hadoop distribution, such as mapr31.

1. Set the usernames in the `<value>` tag for the proxy users as needed. The username you use should be recognized by every node in the MapR cluster.

Component	**Proxy User Property**
HDFS	pentaho.hdfs.proxy.user
Mapreduce	pentaho.mapreduce.proxy.user
Pig	pentaho.pig.proxy.user
Sqoop	pentaho.sqoop.proxy.user
Oozie	pentaho.oozie.proxy.user

Here is an example of modified code.

```
<configuration>
<property>
<name>pentaho.hdfs.proxy.user</name>
<value>jdoe</value>
</property>
<property>
<name>pentaho.mapreduce.proxy.user</name>
<value>bmichaels</value>
</property>
<property>
```

```
<name>pentaho.pig.proxy.user</name>
<value>jdoe</value>
</property>
<property>
<name>pentaho.sqoop.proxy.user</name>
<value>cclarke</value>
</property>
<property>
<name>pentaho.oozie.proxy.user</name>
<value>jdoe</value>
</property>
```

1. Remove comment tags from proxy properties you want to use.
2. Save and close the file.
3. Repeat this process in Spoon. The core-site.xml file is located in data-integration/plugins/pentaho-big-data-plugin/hadoop-configurations/[hadoop distribution].

Set Up Spoofing on Windows Client Node

To set up spoofing on a Windows client node, indicate the spoofed user in the core-site.xml file.

1. On the DI Server node, open the core-site.xml file in pentaho-solutions/system/kettle/plugins/pentaho-big-data-plugin/hadoop-configurations/[hadoop distribution].

[hadoop distribution] is the name of the Hadoop distribution, such as mapr31.

1. Add the following to the file.

```
<property>
  <name>hadoop.spoofed.user.uid</name>
  <value>{UID}</value>
</property>
<property>
  <name>hadoop.spoofed.user.gid</name>
  <value>{GID}</value>
</property>
<property>
  <name>hadoop.spoofed.user.username</name>
  <value>{id of user who has UID}</value>
</property>
```

- Replace **{id of user who has UID}** with the username the principal in the `config.properties` file.
- Replace **{UID}** with the `hadoop.spoofed.user.username` UID.
- Replace **{GID}** with the `hadoop.spoofed.user.username` GID.

1. Save and close the file.
2. Repeat these steps for Spoon. In Spoon the `core-site.xml` file is in `data-integration/plugins/pentaho-big-data-plugin/hadoop-configurations/[hadoop distribution]`.

Apply AES Password Encryption

There are two ways to secure passwords in PDI: Kettle obfuscation and AES. Kettle obfuscation is applied by default. To increase security, use the Advanced Encryption Standard (AES) instead. The password security method you choose is applied to all passwords, including those in database connections, transformation steps, and job entries. To learn more about AES, see http://en.wikipedia.org/wiki/Advanced_Encryption_Standard.

NOTE:

If you switch password security methods, all existing passwords will also use new method.

Create an AES Key File

The key file is a text file that contains the encryption key. You can use 128-bit, 192-bit, or 256-bit encryption strengths. Based on your country of residence, there might be legal restrictions on using the stronger 192-bit or 256-bit encryption strengths. To learn more about legal restrictions, see the Oracle site.

To use the 192-bit or 256-bit encryption strengths, install the Java Cryptography Extension (JCE). You do not need to install the JCE to use 128-bit encryption. To learn more about the JCE, see the Oracle site.

1. Create a text file that contains a key phrase, such as `!@ExampleKey#123`. Note that leading and trailing whitespaces are ignored.
2. Save and close the file.

NOTE:

Safeguard the key file. If the key file becomes corrupted or lost, passwords cannot be decrypted.

Specify AES Variables in kettle.properties

Set AES-specific variables in the `kettle.properties` file for Spoon, the DI Server, and any clusters.

1. Open the kettle.properties file for Spoon. By default, the kettle.properties file is in the user's home directory.
2. Add the following variables and values.

Variable	Description	Value
KETTLE_PASSWORD_ENCODER_PLUGIN	**Required**. Indicates the type of plugin used.	AES
KETTLE_AES_KEY_FILE	**Required**. Indicates the path to the key file.	Path to the key file. Relative paths are resolved against the Kettle working directory,

		NOT the location of the kettle.properties file. Example: c:/securearea/keyfile.txt
KETTLE_AES_KETTLE_PASSWORD_HANDLING	**Optional**. Maintain backwards compatibility by setting this variable to **Decode**. If this is not set, Kettle encoded passwords are not decoded.	DECODE

1. Save and close the `kettle.properties` file.
2. Repeat this process for other `kettle.properties` files on the DI Server and Cluster nodes.
3. You might need to stop and restart Spoon, DI Server, and the Cluster nodes for the `kettle.properties` file to take effect.

Verify Correct Application

After you have applied AES Password encryption, test to make sure it works properly.

1. Start Spoon.
2. Create a blank transformation.
3. Add a database connection that requires a password.
4. Save, then the close the transformation.
5. Use a text editor to open the transformation you just saved, then search for the name of the connection you created.
6. Examine the password. If the password is preceded by the letters `AES`, the encryption method was applied properly.
7. Close the transformation.