pentaho

Version 5.1

Use Manual All Server Installation

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

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

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

Third-Party Open Source Software

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

Contact Us

Global Headquarters Pentaho Corporation Citadel International, Suite 340

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 to the Manual Installation Process

This section explains how to manually install the Pentaho Business Analytics (BA) Server and configure it to run on the database and web application server of your choice. With this installation option you can choose to house the BA Repository on a PostgreSQL, MySQL, or Oracle database. The BA Repository contains solution content, scheduling, and audit tables needed for the BA Server to operate. You can also choose to deploy the BA Server on either the JBoss or Tomcat web application servers. With this installation option, you must supply, install, and configure your chosen database and web application server yourself.

Prerequisites

Read Select BA Installation Option to make sure that this is the best installation option for you. Also, before you begin, check the Supported Technologies tables to make sure that your server computer, BA Repository database, and web browser meet Pentaho's requirements for the current version of the software.

Expertise

The topics in this section are written for IT administrators who know where data is stored, how to connect to it, details about the computing environment, and how to use the command line to issue commands for Microsoft Windows or Linux. You should also know how to install a database and a web application server.

Tools

You must supply a workstation that meets the hardware and software requirements indicated in the Supported Technologies section, as well as a supported operating system and JRE or JDK.

Login Credentials

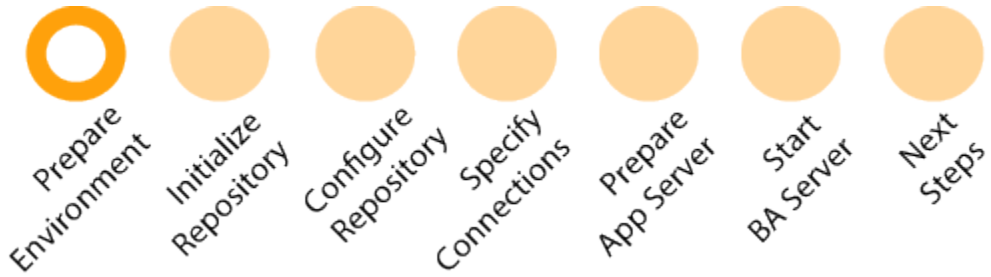
You must be logged onto an account that has administrative privileges to perform the tasks in these sections. Additionally, Linux users need to use the **root** account for some tasks.

Overview of the Installation Process

To install the BA Server, perform the steps indicated in the *guidepost*.

- **Prepare Environment**: Explains how to prepare your computer for software installation.
- **Initialize Repository**: Provides information about how to run DDL scripts that create tables for the BA Repository.
- **Configure Repository**: Provides information about how to configure the BA Repositories on your selected database.
- **Specify Connections**: Explains how to specify the JNDI and JDBC connections to the BA Repository.
- **Prepare Web App Server**: Provides instructions about how to disable unnecessary scans, allot additional memory and time for BA Server deployment, and reference the required Oracle JDK packages.
- **Start BA Server**: Explains how to modify startup files and deploy the BA Server WAR files.
- **Next Steps**: Indicates what to do after the BA Server has been installed.

Prepare Environment



Overview

To prepare the computer on which you plan to install the BA Server, complete these tasks.

Create User Account

Create Windows User Account

If you plan to install on a server that runs Windows, you do not need to create a special user account on the server. However, you should use an account that has administrator privileges to complete the tasks in these instructions.

Create Linux User Account

If you plan to install the BA Server in a Linux environment you must create a user account named **pentaho** on the server computer. By default, license information for Pentaho products is stored in the home directory for this account. Use this account to perform installation tasks that do not require root access. Also, use this account to run start and stop server scripts.

1. Open a **Terminal** window on the server. If you plan to install the BA Server on a remote computer, establish an OpenSSH session to the remote server.
2. In the **Terminal** window, log in as the **root** user by typing this command.

   ```
   su root
   ```

3. When prompted, type the password in the **Terminal** window.
4. In the **Terminal** window, create a new user account called **pentaho**, along with the `pentaho` home directory, by typing this line.

   ```
   sudo useradd -s /bin/bash -m pentaho
   ```

Note: `/bin/bash` indicates that the user account should be created using the Bash shell. In many Linux distributions, the default new user shell is `/bin/sh` or some equivalent, such as Dash, that might not use the `~/.bashrc` configuration file by default. If you don't have or want to use Bash, adjust the instructions throughout in this section accordingly.

5. In the **Terminal** window, assign a password for the **pentaho** user by typing this line.

```
sudo passwd pentaho
```

6. Verify that you can log in using the newly-created **pentaho** user account.
   a. In the **Terminal** window, attempt to log in by typing this line.

```
su pentaho -
```

   b. Type the password for the **pentaho** user account if you are prompted.
   c. Use the **Terminal** window to navigate to the pentaho directory to verify that it has been created. By default, it is in the `/home` directory.
   d. Close the **Terminal** window.

Create Directory Structure

Create Windows Directory Structure

1. Log into the machine on which you will run the BA Server.
2. Create these directory paths.

```
pentaho\server\biserver-ee
<your home directory>\.pentaho
```

3. Verify that you have the appropriate permissions to read, write, and execute commands in the directories you created.
   a. Open **Windows Explorer** and right-click the `pentaho` directory.
   b. Select the **Properties** option and the **Security** tab to verify that you have read, write, and execute permissions.
   c. In **Windows Explorer** navigate to `server` directory, then right-click.
   d. Select the **Properties** option and the **Security** tab to verify that you have read, write, and execute permissions.
   e. In **Windows Explorer** and navigate to the `biserver-ee` directory and right-click it.
   f. Select the **Properties** option and the **Security** tab to verify that you have read, write, and execute permissions.
   g. In **Windows Explorer** and navigate to the `.pentaho` directory that is in your home directory and right-click it.
   h. Select the **Properties** option and the **Security** tab to verify that you have read, write, and execute permissions.

Create Linux Directory Structure

1. Log into the machine on which you will run the BA Server. Make sure that you are logged in as the **pentaho** user.
2. Create this directory path from home directory (`pentaho`).

```
<your home directory>/pentaho/server/biserver-ee
<your home directory>/.pentaho
```

3. Verify that you have the appropriate permissions to read, write, and execute commands in the directories you created.
    a. In Linux check the permissions of the `pentaho`, `server`, and `biserver-ee` directories by opening a **Terminal** window, navigating to the `pentaho` directory, then typing this command.

```
ls -ld ../.pentaho ../pentaho ../pentaho/server ../pentaho/server/
biserver-ee
```

    b. Make sure that permissions for the directories allow you to read, write, and execute in those directories.

Install the Web Application Server

The BA Server can be deployed on either the Tomcat or JBoss web application server. By default, BA Server software is configured for Tomcat. This means that if you choose to use Tomcat, you will need to make fewer configuration changes than you would if you choose to use JBoss.

You must install the web application server yourself. If you already have a Tomcat or JBoss web application server installed and you want to deploy the BA Server on it, please skip this step.

1. To download and install the web application software, use the instructions in the documentation for the web application server of your choice. We recommend that you install the web application server in the `pentaho/server/biserver-ee` directory.
2. Verify the web application server is installed correctly by starting it and viewing the default page. If the web application server does not start, troubleshoot it using the web application server's documentation before you continue with the BA Server installation process.
3. Stop the web application server.

Install the BA Repository Host Database

The BA Repository houses data needed for Pentaho tools to provide scheduling and security functions, as well as metadata and models for reports that you create.

You can choose to host the BA Repository on the PostgreSQL, MySQL, or Oracle database. By default, Pentaho software is configured to use the PostgreSQL Database. If you already have a BA Repository database installed, you can skip this step. To install for the host database for the BA Respository, do these things.

1. To download and install the BA Repository database, use the instructions in the documentation for the database of your choice. It does not matter where you install the database.
2. Verify that the BA Repository database is installed correctly. You can do this by connecting to your database and viewing the contents of any default databases that might have been created upon

installation. Consult the user documentation for the database that you installed for further details.
Note: If you are not familiar with SQL, consider using a visual database design tool to connect to the database and view its contents.

- PGAdminIII is bundled with PostgreSQL.
- MySQL Workbench can be used with MySQL. It is available as a separate download. Check the MySQL site for details on how to obtain this design tool.
- Oracle SQL Developer can be used with Oracle. It is available as a separate download. Check the Oracle site for details on how to obtain this design tool.

Install Java JRE or JDK

Make sure that the version of the Java Runtime Environment (JRE) or the Java Development Kit (JDK) that Pentaho needs to run is installed on your system. You do not need to uninstall other versions of Java if you already have them running on your system. These instructions explain how to check for your default version of Java that is running on your computer and where to get the required version if you need one.

1. Check the supported technologies list to see which version of the JRE or JDK is needed for the software.
2. If you have not done so already, log into the computer on which you plan to install the software. Ensure that you have the appropriate permissions to install software.
3. Open a **Terminal** or a **Command Prompt** window. Enter this command.

   ```
   java -version
   ```

4. If the version of Java does not match the version needed to run Pentaho software, check your system to see if there are other versions of Java installed.
5. If the version of Java that you need to run Pentaho software is not on your system, download it from the Oracle site and install it.

Download and Unpack Installation Files

If you want to install specific Pentaho software, obtain the installation packages from the Pentaho Customer Support Portal. Consult your Welcome Kit if you need more information about the portal. The Pentaho BA Server software, data files, and examples are stored in pre-packaged `.war` and `.zip` files. Manually copy these files to correct directories.
Note: These instructions explain how to install the BA Server only. If you want to install the plugins, finish installing the BA Server, then see Install Only BA Tools.

1. Make sure the web application server on which you plan to deploy the BA Server has been stopped.
2. Download the `biserver-manual-ee-5.x.x-dist.zip` file.
3. Unpack the file by completing these steps.
   a. Use a zip tool to extract the distribution files you just downloaded.
   b. Open a **Command Prompt** or **Terminal** window and navigate to the folder that contains the files you just extracted.
   c. Enter one of the following at the prompt.
   Windows:

   ```
   install.bat
   ```

Linux:

```
./install.sh
```

d. Read the license agreement that appears. Select **I accept the terms of this license agreement**, then click **Next**. **Note:** If you are unpacking the file in a non-graphical environment, open a **Terminal** or **Command Prompt** window and type `java -jar installer.jar -console` and follow the instructions presented in the window.

e. Indicate where you want the file to be unpacked. It doesn't matter where because you will be manually placing the files in the appropriate directories later in these instructions.

f. Click the **Next** button.

g. The **Installation Progress** window appears. Progress bars indicate the status of the installation. When the installation progress is complete, click **Quit** to exit the Unpack Wizard.

4. Navigate to the directory where you unpacked the files. Copy the `pentaho.war` and `pentaho-style.war` files to the appropriate directory. The directory you choose is determined by the web application server installed.
   - **Tomcat:** `pentaho/server/biserver-ee/<your tomcat installation directory>/webapps`
   - **JBoss:** `pentaho/server/biserver-ee/<your jboss installation directory>/standalone/deployments`
7. Extract the `pentaho-solutions.zip` file to the `pentaho/server/biserver-ee` subdirectory. After you've extracted the zip file, the `pentaho-solutions` directory appears in the `pentaho/server/biserver-ee` directory.
8. Extract the `pentaho-data.zip` file to the `pentaho/server/biserver-ee subdirectory`. After you've extracted the zip file, a `data` directory appears in the `biserver-ee` directory.
9. Extract the `license-installer.zip` file to the `pentaho/server` directory. After you've extracted the zip file, the `license-installer` directory appears in the `pentaho/server` directory.
10. If you unzipped the plugin files, copy the extracted plugin folders into the `pentaho-solutions/system` folder.
11. Verify that the files have been placed in the following places by comparing the following directory structure with yours. **Note:** If your web application server is not in the `pentaho/server/biserver-ee` directory, the `pentaho.war` and `pentaho-style.war` files should appear where you've chosen to install your web application server.

Tomcat File Locations:

- `<your home directory>/.pentaho`
- `pentaho/server/license-installer`
- `pentaho/server/biserver-ee/<your tomcat installation directory>/webapps/pentaho.war`
- `pentaho/server/biserver-ee/<your tomcat installation directory>/webapps/pentaho.war`
- `pentaho/server/biserver-ee/<your tomcat installation directory>/webapps/pentaho-style.war`
- `pentaho/server/biserver-ee/data`
- `pentaho/server/biserver-ee/pentaho-solutions`
- `pentaho/server/biserver-ee/pentaho-solutions/systems/default-content/samples-5.x.zip`

JBoss File Locations:

- `<your home directory>/.pentaho`
- `pentaho/server/license-installer`

- pentaho/server/biserver-ee/<your jboss installation directory>/standalone/deployments/pentaho.war
- pentaho/server/biserver-ee/<your jboss installation directory>/standalone/deployments/pentaho-style.war
- pentaho/server/biserver-ee/data
- pentaho/server/biserver-ee/pentaho-solutions
- pentaho/server/biserver-ee/pentaho-solutions/system/default-content/samples-5.x.zip

Set Environment Variables

Set the PENTAHO_JAVA_HOME variable to indicate the path to the Java JRE or JDK that Pentaho should use. Also, set the PENTAHO_INSTALLED_LICENSE_PATH variable so that when you start Pentaho, the licenses can install. If you do not set these variables, Pentaho will not start correctly. To set environment variables, you should be logged into an account that has administrator-level privileges. For Linux systems, you must be logged into the **root** user account.

Set Windows PENTAHO_JAVA_HOME and PENTAHO_INSTALLED_LICENSE_PATH Variables

1. Open a **Command Prompt**.
2. At the prompt, set the path of the PENTAHO_JAVA_HOME variable to the path of your Java 7 installation. Here is an example.

```
SET PENTAHO_JAVA_HOME=C:\Program Files\Java\jre7
```

3. At the prompt, set the path of the PENTAHO_INSTALLED_LICENSE_PATH variable to the .pentaho directory. Here is an example.

```
SET PENTAHO_INSTALLED_LICENSE_PATH=C:\Users\jdoe\.pentaho
```

4. Log out, then log back in.
5. To verify that the variable has been properly set, open a **Command Prompt** window, then type this.

```
echo %PENTAHO_JAVA_HOME% %PENTAHO_INSTALLED_LICENSE_PATH%
```

6. The path that you entered for the Java 7 installation should appear. If it does not, try to set the environment variable again.

Set Linux PENTAHO_JAVA_HOME and PENTAHO_INSTALLED_LICENSE_PATH Variables

1. Open a **terminal** window and log in as **root**.
2. Open the /etc/environment file with a text editor. **Note:**The vi and gedit text editors are available on most Linux machines. For example, to open the /etc/environment file with gedit, type this.

```
gedit /etc/environment
```

3. Indicate where you installed Java in your /etc/environment file by typing this. **Note:** Substitute /usr/lib/jvm/java-7-sun with the location of the JRE or JDK you installed on your system.

```
export PENTAHO_JAVA_HOME=/usr/lib/jvm/java-7-sun
```

4. Indicate the location of the `.pentaho` directory by typing this.

```
export PENTAHO_INSTALLED_LICENSE_PATH=/<your home folder>/.pentaho
```

5. Save and close the file.
6. Log out, then log back in for the change to take effect.
7. Verify that the `PENTAHO_JAVA_HOME` variable is properly set by opening a **Terminal** window and typing this.

```
env | grep PENTAHO_JAVA_HOME
```

8. The path to the variable should appear. If it does not, try setting the environment variable again.
9. Verify that the `PENTAHO_INSTALLED_LICENSE_PATH` variable is properly set by opening a **Terminal** window and typing this.

```
env | grep PENTAHO_INSTALLED_LICENSE_PATH
```

10. The path to the variable should appear. If it does not, try setting the environment variable again.

Advanced Topics

Complete the instructions in this section only if you have a headless node or if you plan to install on a Mac OS.

Prepare a Headless Linux or Solaris Server

There are two headless server scenarios that require special procedures on Linux and Solaris systems. One is for a system that has no video card; the other is for a system that has a video card, but does not have an X server installed. In some situations -- particularly if your server doesn't have a video card -- you will have to perform both procedures in order to properly generate reports with the BA Server.

Systems without video cards

The **java.awt.headless** option enables systems without video output and/or human input hardware to execute operations that require them. To set this application server option when the BA Server starts, you will need to modify the startup scripts for either the BA Server, or your Java application server. You do not need to do this now, but you will near the end of these instruction when you perform the Start BA Server step. For now, add the following item to the list of CATALINA_OPTS parameters: **-Djava.awt.headless=true**.

The entire line should look something like this:

```
export CATALINA_OPTS="-Djava.awt.headless=true -Xms4096m -Xmx6144m -
XX:MaxPermSize=256m -Dsun.rmi.dgc.client.gcInterval=3600000 -Dsun.rmi.dgc.server.
gcInterval=3600000"
```

If you intend to create a BA Server service control script, you must add this parameter to that script's CATALINA_OPTS line.
Note: If you do not have an X server installed, you must also follow the below instructions.

Systems without X11

To generate charts, the Pentaho Reporting engine requires functionality found in X11. If you are unwilling or unable to install an X server, you can install the **xvfb** package instead. xvfb provides X11 framebuffer emulation, which performs all graphical operations in memory instead of sending them to the screen.

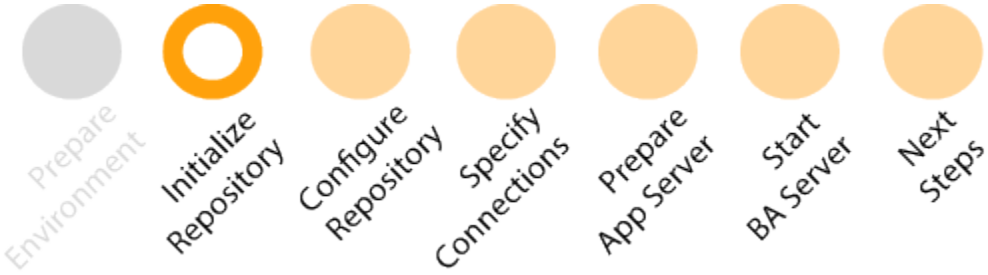
Use your operating system's package manager to properly install xvfb.

Adjust Amount of Memory Mac OS Allocates for PostgreSQL

If you plan to install the software on a Mac OS, and you choose to use PostgreSQL, you need to increase the amount of memory that the Mac OS allocates for PostgreSQL. You can skip these instructions if you plan to install the software on Windows or Linux.

PostgreSQL is the name of the default database that contains audit, schedule and other data that you create. PostgreSQL starts successfully only if your computer has allocated enough memory. Go to http://www.postgresql.org/docs/devel/static/kernel-resources.html and follow the instructions there on how to adjust the memory settings on your computer.

Initialize Repository



Before you prepare the BA Repository complete the tasks in Prepare Environment.

Pentaho stores content about reports that you create, examples we provide, report scheduling data, and audit data in the BA Repository. The BA Repository resides on the database that you installed during the Prepare Environment step. The BA Repository consists of three repositories: *Jackrabbit*, *Quartz*, and *Hibernate*.

- *Jackrabbit* contains the solution respository, examples, security data, and content data from reports that you use Pentaho software to create.
- *Quartz* holds data that is related to scheduling reports and jobs.
- *Hibernate* holds data that is related to audit logging.

This step only consists of one task: Initialize the database. In this task you run DDLs that contain SQL commands that create the Jackrabbit, Quartz, and Hibernate databases, as well as the Operations Mart schema.

Initialize PostgreSQL BA Repository Database

To initialize PostgreSQL so that it serves as the BA Repository, run SQL scripts to create the Hibernate, Quartz, and Jackrabbit (also known as the JCR) databases.

NOTE:

Your PostgreSQL configuration must support logins from all users. This is not always the default configuration, so you may have to edit your `pg_hba.conf` file to support this option. If you do need to make changes to pg_hba.conf, you must restart the PostgreSQL server before proceeding.

1. To make the databases that you create more secure, Pentaho recommends that you change the default passwords in the SQL script files to ones that you specify. If you are evaluating Pentaho, you might want to skip this step. If you do decide to make the databases more secure, use a text editor to change the passwords in these files:
   - `pentaho/server/biserver-ee/data/postgresql/create_jcr_postgresql.sql`
   - `pentaho/server/biserver-ee/data/postgresql/create_quartz_postgresql.sql`
   - `pentaho/server/biserver-ee/data/postgresql/create_repository_postgresql.sql`

   Here is an example of a password change made in the `create_jcr_postgresql.sql` file.

```
CREATE USER jcr_user PASSWORD 'myNewPassword'
```

5. **Windows:** The commands you use to run the SQL scripts depends on your operating system. For windows, do this.
    a. Open a **SQL Shell** window. The **SQL Shell** window is installed with PostgreSQL.
    b. When prompted for the server enter the name of the server if you are not using the default (localhost). If you are using the default, do not type anything and press Enter.
    c. When prompted for the database enter the name of the database if you are not using the default (postgres) If you are using the default, do not type anything and press Enter.
    d. When prompted for the port enter the name of the port if you are not using the default (5432). If you are using the default port, do not type anything and press Enter.
    e. When prompted for the username, accept the default, then press Enter.
    f. When prompted for the password, enter the password that you indicated when you installed PostgreSQL.
    g. Run the script to create the Jackrabbit database by typing this. (If necessary, change `\pentaho\server\biserver-ee` to the place where you unpacked your pentaho files.)

    ```
    \i /pentaho/server/biserver-ee/data/postgresql/create_jcr_postgresql.
    sql
    ```

    h. Run the script to create the hibernate database by typing this. (If necessary, change `\pentaho\server\biserver-ee` to the place where you unpacked your pentaho files.)

    ```
    \i /pentaho/server/biserver-ee/data/postgresql/create_repository_
    postgresql.sql
    ```

    i. Run the script to create the Quartz database by typing this. (If necessary, change `\pentaho\server\biserver-ee` to the place where you unpacked your pentaho files.)

    ```
    \i /pentaho/server/biserver-ee/data/postgresql/create_quartz_
    postgresql.sql
    ```

    Enter the password at the prompt. The default is **password**.

    j. To switch to the Hibernate database, type this.

    ```
    \c postgres
    ```

    k. Run the script to create the Operations Mart database by typing this. (If necessary, change `\pentaho\server\biserver-ee` to the place where you unpacked your pentaho files.)

    ```
    \i /pentaho/server/biserver-ee/data/postgresql/pentaho_mart_
    postgresql.sql
    ```

    l. Exit from the window by pressing the **CTRL** + **C** keys.
6. **Linux:** To run the SQL scripts on a Linux system, do this.
    a. Open a **Terminal** window. You should be logged in as the **pentaho** user.
    b. Sign into PostgreSQL by typing `psql -U postgres -h localhost` at the prompt.

c. Run the script to create the Jackrabbit database by typing this. (If necessary, change `/pentaho/server/biserver-ee` to the place where you unpacked your pentaho files.)

```
\i ~/pentaho/server/biserver-ee/data/postgresql/create_jcr_postgresql.sql
```

Note: If necessary, change the `~/pentaho/server/biserver-ee` directory path to the place where you unpacked your pentaho files.

d. Run the script to create the hibernate database by typing this. (If necessary, change `/pentaho/server/biserver-ee` to the place where you unpacked your pentaho files.)

```
\i ~/pentaho/server/biserver-ee/data/postgresql/create_repository_postgresql.sql
```

Note: If necessary, change the `~/pentaho/server/biserver-ee` to the place where you unpacked your pentaho files.

e. Run the script to create the Quartz database by typing this. (If necessary, change `/pentaho/server/biserver-ee` to the place where you unpacked your pentaho files.)

```
\i ~/pentaho/server/biserver-ee/data/postgresql/create_quartz_postgresql.sql
```

Note: If necessary, change the `~/pentaho/server/biserver-ee` directory path to the place where you unpacked your pentaho files.

f. To switch to the Hibernate database, type this.

```
\c postgres
```

g. Run the script to create the Operations Mart database by typing this.

```
\i ~/pentaho/server/biserver-ee/data/postgresql/pentaho_mart_postgresql.sql
```

h. Exit from the window by pressing the **CTRL** + **C** keys.

7. To verify that databases and user roles have been created, do this.
   a. Open the pgAdminIII tool. pgAdminIII is bundled with both the Windows and Linux versions of PostgreSQL.
   b. To view the contents of PostgreSQL, click the PostgreSQL folder in the **Object Browser**, then enter the password when prompted.
   c. In the **Object Browser**, click the **Databases** folder. The Jackrabbit, Postgres, Hibernate and Quartz databases should appear.
   d. In the **Object Browser**, click the **Login Roles** folder. The jcr_user, pentaho_user, hibuser, and postgres user accounts appear.
   e. If the databases and login roles do not appear, go to the beginning of these instructions and try running the scripts again.
   f. Select **File > Exit** to exit from pgAdminIII.

Initialize MySQL BA Repository Database

To initialize MySQL so that it serves as the BA Repository, run SQL scripts to create the Hibernate, Quartz and Jackrabbit (also known as the JCR) databases.

Note: Use the ASCII character set when you run these scripts. Do not use UTF-8 because there are text string length limitations that might cause the scripts to fail.

1. To make the databases that you create more secure, Pentaho recommends that you change the default passwords in the SQL script files to ones that you specify. If you are evaluating Pentaho, you might want to skip this step. If you do decide to make the databases more secure, use a text editor to change the passwords in these files:
   - `pentaho/server/biserver-ee/data/mysql5/create_jcr_mysql.sql`
   - `pentaho/server/biserver-ee/data/mysql5/create_quartz_mysql.sql`
   - `pentaho/server/biserver-ee/data/mysql5/create_repository_mysql.sql`

   Here is an example of a password change made in the `create_jcr_mysql.sql` file.

   ```
   grant all on jackrabbit.* to 'jcr_user'@'localhost' identified by
   'myNewPassword';
   ```

5. The process for running the SQL scripts against MySQL are the same for both Windows and Linux machines.
   a. Run the `create_quartz_mysql.sql` script in the **Terminal** or **Command Prompt** window by typing: `mysql -u root -p < create_quartz_mysql.sql.`
   b. Run the `create_repository_mysql.sql` script in the **Terminal** or **Command Prompt** window by typing: `mysql -u root -p < create_repository_mysql.sql.`
   c. Run the `create_jcr_mysql.sql` script in the **Terminal** or **Command Prompt** window by typing: `mysql -u root -p < create_jcr_mysql.sql.`
   d. Run the `pentaho_mart_mysql.sql` script in the **Terminal** or **Command Prompt** window by typing: `mysql -u root -p < pentaho_mart_mysql.sql.`
6. To verify that databases and user roles have been created, do this.
   a. Open the **MySQL Workbench** tool. **MySQL Workbench** is freely available at the MySQL development site.
   b. Make sure that the Jackrabbit (JCR), Hibernate, and Quartz databases are present.
   c. Make sure that the jcr_user, hibuser, and pentaho_user user accounts are present.
   d. If the databases and login roles do not appear, go to the beginning of these instructions and try running the scripts again.
   e. Exit from the **MySQL Workbench**.

Initialize Oracle BA Repository Database

To initialize Oracle so it serves as the BA Repository, run SQL scripts to create the Hibernate, Quartz and Jackrabbit (also known as the JCR) databases.

1. To make the databases that you create more secure, Pentaho recommends that you change the default passwords in the SQL script files to ones that you specify. If you are evaluating Pentaho, you might want to skip this step. If you do decide to make the databases more secure, use a text editor to change

the passwords in these files. (Also, for each file, edit the **datafile** path with the path to your Oracle installation.)

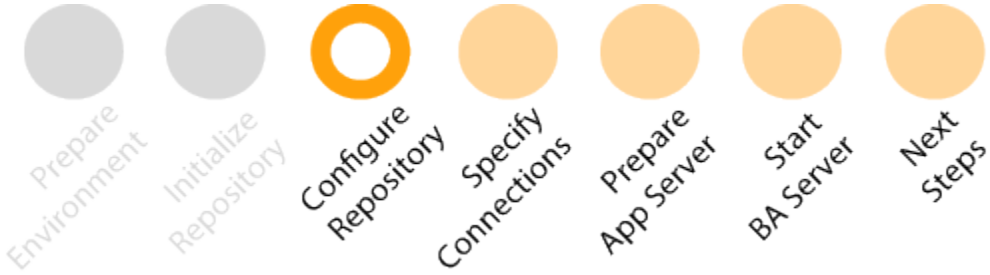
- `pentaho/server/biserver-ee/data/oracle10g/create_jcr_ora.sql`
- `pentaho/server/biserver-ee/data/oracle10g/create_quartz_ora.sql`
- `pentaho/server/biserver-ee/data/oracle10g/create_repository_ora.sql`

Here is an example of a password change made in the `create_jcr_ora.sql` file.

```
--conn admin/myNewPassword@pentahocreate user jcr_user
identified by "myNewPassword" default tablespace pentaho_
tablespace quota unlimited on pentaho_tablespace temporary
tablespace temp quota 5M on system;
```

5. Although there are several different methods for running SQL scripts, these instructions explain how to run SQL*Plus from a **Terminal** or **Command Prompt** window. These instructions are the same for both Windows and Linux. If you prefer to run SQL scripts using another method, modify instructions accordingly.
    a. Open a **Terminal** or **Command Prompt** window, start the **SQL*Plus** and log in.
    b. Run the script to create the Jackrabbit database by typing `START create_jcr_ora`. If necessary, append the path to the `create_jcr_ora.sql` path in the command.
    c. Run the script to create the repository database by typing `START create_repository_ora`. If necessary, append the path to the `create_repository_ora.sql` path in the command.
    d. Run the script to create the Quartz database and users by typing `START create_quartz_ora`. If necessary, append the path to the `create_quartz_ora.sql` path in the command.
    e. Run the script to create the Operations Mart database and users by typing `START pentaho_mart_ora`. If necessary, append the path to the `pentaho_mart_ora.sql` path in the command.
6. To verify that databases and user roles have been created, do this.
    a. In the **Terminal** or **Command Prompt** window that is running **SQL*Plus**, make sure that the Jackrabbit database has been created by typing DESCRIBE JACKRABBIT;. The column definitions should appear when you press **Enter**.
    b. Make sure the Quartz database has been created by typing DESCRIBE QUARTZ;. The column definitions for the Quartz table should appear when you press **Enter**.
    c. To see the users that have been created, type SELECT USERNAME FROM DBA_USERS.
    d. If the databases and login roles do not appear, go to the beginning of these instructions and try running the scripts again.
    e. Exit from **SQL*Plus**.

Configure Repository



Before you configure the BA Repository, complete the tasks in Initialize Repository.

Tasks performed during this step include configuring Audit, Quartz, and Hibernate properties. Tasks are grouped by the BA Repository database you have.

Configure PostgreSQL BA Repository Database

These instructions explain how to configure Quartz, Hibernate, Jackrabbit, and Pentaho Security for use with the PostgreSQL database. By default, the files edited in this section are configured for a PostgreSQL database that runs on port 5432. The default password is also in these files. If you have a different port, different password, or if had the system configured using a different database and now you want to change it back to PostgreSQL, complete all of the instructions in these steps.

Configure Quartz on PostgreSQL BA Repository Database

When you use Pentaho to schedule an event, such as a report to be run every Sunday at 1:00 a.m. EST, event information is stored in the Quartz JobStore. During the installation process, you must indicate where the JobStore is located. To do this, modify the `quartz.properties` file.

1. Open the `pentaho/server/biserver-ee/pentaho-solutions/system/quartz/quartz.properties` file in the text editor of your choice.
2. Make sure that in the `#_replace_jobstore_properties` section of the file, the `org.quartz.jobStore.driverDelegateClass` is set to `org.quartz.impl.jdbcjobstore.PostgreSQLDelegate`.
3. In the `# Configure Datasources` section of the file, set the `org.quartz.dataSource.myDS.jndiURL` equal to `Quartz`, like this.

```
org.quartz.dataSource.myDS.jndiURL = Quartz
```

4. Save the file and close the text editor.

Configure Hibernate Settings for PostgreSQL BA Repository Database

Modify the hibernate settings file to specify where Pentaho will find the BA Repository's hibernate configuration file. The hibernate configuration file specifies driver and connection information, as well as dialects and how to handle connection closes and timeouts.

1. Open `pentaho/server/biserver-ee/pentaho-solutions/system/hibernate/hibernate-settings.xml` in a text editor.
2. Verify that the location of the PostgreSQL hibernate configuration file appears. Make changes if necessary.

   ```
   <config-file>system/hibernate/postgresql.hibernate.cfg.xml</config-file>
   ```

3. Save the file if you had to make changes, then close it. Otherwise, just close it.
4. Open `pentaho/server/biserver-ee/pentaho-solutions/system/hibernate/postgresql.hibernate.cfg.xml` in a text editor.
5. Make sure that the password and port number match the ones you specified in your configuration. Make changes as necessary, then save and close the file.

Modify Jackrabbit BA Repository Information for PostgreSQL

Indicate which database houses the BA Repository as well as the port, url, username, and password. All of the information needed to configure the repository for the PostgreSQL, MySQL, and Oracle BA Repository databases appear. By default, the PostgreSQL sections are not commented out, but the MySQL and Oracle sections are. To modify this file so that it works for your BA Repository, you will need to make sure that the sections that refer to your BA Repository database are not commented out, and the sections refer to other BA Repository databases are commented out.

When code is commented out, it appears between the `<!--` and `-->` tags. The information in between the tags is commented out, and is therefore not executed by the software. In this example, the code `<!` `<FileSystem class="org.apache.jackrabbit.core.fs.db.OracleFileSystem">` is commented out.

```
<!--
<FileSystem class="org.apache.jackrabbit.core.fs.db.OracleFileSystem">
-->
```

To make sure that the Jackrabbit repository is set so that PostgreSQL is the default database, do this.

1. Use a text editor to open the `pentaho/server/biserver-ee/pentaho-solutions/system/jackrabbit/repository.xml` file.
2. In the Repository's FileSystem part of the code, make sure that the PostgreSQL lines of code are not commented out, but the Oracle and MySQL lines are. The code should look like this.

   ```
   <!--
     <FileSystem class="org.apache.jackrabbit.core.fs.db.DbFileSystem">
       <param name="driver" value="com.mysql.jdbc.Driver"/>
   ```

```
    ...
    <param name="schemaObjectPrefix" value="fs_repos_"/>
  </FileSystem>
  <FileSystem class="org.apache.jackrabbit.core.fs.db.OracleFileSystem">
    <param name="url" value="jdbc:oracle:thin:@localhost:1521:orcl"/>
    ...
    <param name="schemaObjectPrefix" value="fs_repos_"/>
    <param name="tablespace" value="jackrabbit"/>
  </FileSystem>
-->
  <FileSystem class="org.apache.jackrabbit.core.fs.db.DbFileSystem">
    <param name="driver" value="org.postgresql.Driver"/>
    <param name="url" value="jdbc:postgresql://localhost:5432/jackrabbit"/>
    <param name="user" value="jcr_user"/>
    <param name="password" value="password"/>
    <param name="schema" value="postgresql"/>
    <param name="schemaObjectPrefix" value="fs_repos_"/>
  </FileSystem>
```

NOTE:

If you changed your password when you initialized the database during the Prepare Environment step, or if your database is on a different port, edit the url and password parameters accordingly.

3. In the DataStore section of the code, verify that the PostgreSQL lines of code are not commented out, but the Oracle and MySQL lines are. The code should look like this.

```
<!--
  <DataStore class="org.apache.jackrabbit.core.data.db.DbDataStore">
    <param name="url" value="jdbc:mysql://localhost:3306/jackrabbit"/>
   ...
    <param name="schemaObjectPrefix" value="ds_repos_"/>
  </DataStore>
  <DataStore class="org.apache.jackrabbit.core.data.db.DbDataStore">
    <param name="url" value="jdbc:oracle:thin:@localhost:1521:orcl"/>
   ...
    <param name="schemaObjectPrefix" value="ds_repos_"/>
  </DataStore>
  -->
  <DataStore class="org.apache.jackrabbit.core.data.db.DbDataStore">
    <param name="url" value="jdbc:postgresql://localhost:5432/jackrabbit"/>
    <param name="driver" value="org.postgresql.Driver"/>
    <param name="user" value="jcr_user"/>
```

```
    <param name="password" value="password"/>
    <param name="databaseType" value="postgresql"/>
    <param name="minRecordLength" value="1024"/>
    <param name="maxConnections" value="3"/>
    <param name="copyWhenReading" value="true"/>
    <param name="tablePrefix" value=""/>
    <param name="schemaObjectPrefix" value="ds_repos_"/>
  </DataStore>
```

Note: If you changed your password when you initialized the database during the Prepare Environment step, or if your database is on a different port, edit the url and password parameters accordingly.

4. In the Workspaces section of the code, make sure that the PostgreSQL lines of code are not commented out, but the Oracle and MySQL lines are. This code should look like this.

```
<!--
    <FileSystem class="org.apache.jackrabbit.core.fs.db.DbFileSystem">
      <param name="driver" value="com.mysql.jdbc.Driver"/>
     ...
      <param name="schemaObjectPrefix" value="fs_ws_"/>
    </FileSystem>
    <FileSystem class="org.apache.jackrabbit.core.fs.db.OracleFileSystem">
      <param name="url" value="jdbc:oracle:thin:@localhost:1521:orcl"/>
      ...
      <param name="schemaObjectPrefix" value="fs_ws_"/>
      <param name="tablespace" value="jcr_user"/>
    </FileSystem>
-->
    <FileSystem class="org.apache.jackrabbit.core.fs.db.DbFileSystem">
      <param name="driver" value="org.postgresql.Driver"/>
      <param name="url" value="jdbc:postgresql://localhost:5432/
jackrabbit"/>
      <param name="user" value="jcr_user"/>
      <param name="password" value="password"/>
      <param name="schema" value="postgresql"/>
      <param name="schemaObjectPrefix" value="fs_ws_"/>
    </FileSystem>
```

Note: If you changed your password when you initialized the database during the Prepare Environment step, or if your database is on a different port, edit the url and password parameters accordingly.

5. In the Persistence Manager section of the code, verify that the PostgreSQL lines of code are not commented out, but the Oracle and MySQL lines are. The code should look like this.

```
<!--
    <PersistenceManager class="org.apache.jackrabbit.core.persistence.
bundle.MySqlPersistenceManager">
      <param name="url" value="jdbc:mysql://localhost:3306/jackrabbit"/>
      ...
      <param name="schemaObjectPrefix" value="${wsp.name}_pm_ws_"/>
    </PersistenceManager>
    <PersistenceManager class="org.apache.jackrabbit.core.persistence.
bundle.OraclePersistenceManager">
      <param name="url" value="jdbc:oracle:thin:@localhost:1521:orcl"/>
      ...
      <param name="schemaObjectPrefix" value="${wsp.name}_pm_ws_"/>
      <param name="tablespace" value="jackrabbit"/>
    </PersistenceManager>
   -->
    <PersistenceManager class="org.apache.jackrabbit.core.persistence.
bundle.PostgreSQLPersistenceManager">
      <param name="url" value="jdbc:postgresql://localhost:5432/
jackrabbit"/>
      <param name="driver" value="org.postgresql.Driver"/>
      <param name="user" value="jcr_user"/>
      <param name="password" value="password"/>
      <param name="schema" value="postgresql"/>
      <param name="schemaObjectPrefix" value="${wsp.name}_pm_ws_"/>
    </PersistenceManager>
```

Note: If you changed your password when you initialized the database during the Prepare Environment step, or if your database is on a different port, edit the url and password parameters accordingly.

6. In the Versioning section of the code, verify that the PostgreSQL lines of code are not commented out, but the MySQL and Oracle lines are. The code should look like this.

```
<!--
    <FileSystem class="org.apache.jackrabbit.core.fs.db.DbFileSystem">
      <param name="driver" value="com.mysql.jdbc.Driver"/>
     ...
      <param name="schemaObjectPrefix" value="fs_ver_"/>
    </FileSystem>
    <FileSystem class="org.apache.jackrabbit.core.fs.db.OracleFileSystem">
      <param name="url" value="jdbc:oracle:thin:@localhost:1521:orcl"/>
     ...
      <param name="schemaObjectPrefix" value="fs_ver_"/>
      <param name="tablespace" value="jackrabbit"/>
```

```
    </FileSystem>
    -->
    <FileSystem class="org.apache.jackrabbit.core.fs.db.DbFileSystem">
      <param name="driver" value="org.postgresql.Driver"/>
      <param name="url" value="jdbc:postgresql://localhost:5432/
jackrabbit"/>
      <param name="user" value="jcr_user"/>
      <param name="password" value="password"/>
      <param name="schema" value="postgresql"/>
      <param name="schemaObjectPrefix" value="fs_ver_"/>
    </FileSystem>
```

Note: If you changed your password when you initialized the database during the Prepare Environment step, or if your database is on a different port, edit the url and password parameters accordingly.

7. In the Persistence Manager section of the code that is near the end of the file, verify that PostgreSQL lines of code are not commented out, but the MySQL and Oracle lines are. The codes should look like this.

```
<!--
    <PersistenceManager class="org.apache.jackrabbit.core.persistence.
bundle.MySqlPersistenceManager">
      <param name="url" value="jdbc:mysql://localhost:3306/jackrabbit"/>
     ...
      <param name="schemaObjectPrefix" value="pm_ver_"/>
    </PersistenceManager>

    <PersistenceManager class="org.apache.jackrabbit.core.persistence.
bundle.OraclePersistenceManager">
      <param name="url" value="jdbc:oracle:thin:@localhost:1521:orcl"/>
     ...
      <param name="schemaObjectPrefix" value="pm_ver_"/>
      <param name="tablespace" value="jackrabbit"/>
    </PersistenceManager>
    -->
    <PersistenceManager class="org.apache.jackrabbit.core.persistence.
bundle.PostgreSQLPersistenceManager">
      <param name="url" value="jdbc:postgresql://localhost:5432/
jackrabbit"/>
      <param name="driver" value="org.postgresql.Driver"/>
      <param name="user" value="jcr_user"/>
      <param name="password" value="password"/>
      <param name="schema" value="postgresql"/>
```

```
      <param name="schemaObjectPrefix" value="pm_ver_"/>
    </PersistenceManager>
```

Note: If you changed your password when you initialized the database during the Prepare Environment step, or if your database is on a different port, edit the url and password parameters accordingly.

8. Close and save the file.

Prepare MySQL BA Repository Database

These instructions explain how to configure Quartz, Hibernate, Jackrabbit, and Pentaho Security for a MySQL database. By default, the examples in this section are for a MySQL database that runs on port 3306. The default password is also in these examples. If you have a different port, different password complete all of the instructions in these steps.

Configure Quartz on MySQL BA Repository Database

When you use Pentaho to schedule an event, such as a report to be run every Sunday at 1:00 a.m. EST, event information is stored in the Quartz JobStore. During the installation process, you must indicate where the JobStore is located. To do this, modify the `quartz.properties` file.

1. Open the `pentaho/server/biserver-ee/pentaho-solutions/system/quartz/quartz.properties` file in the text editor of your choice.
2. In the `#_replace_jobstore_properties` section of the file, set the `org.quartz.jobStore.driverDelegateClass` equal to `org.quartz.impl.jdbcjobstore.StdJDBCDelegate`.

```
org.quartz.jobStore.driverDelegateClass = org.quartz.impl.jdbcjobstore.
StdJDBCDelegate
```

3. Save the file and close the text editor.

Configure Hibernate Settings for MySQL

Modify the hibernate settings file to specify where Pentaho should find the BA Repository's hibernate configuration file. The hibernate configuration file specifies driver and connection information, as well as dialects and how to handle connection closes and timeouts.

1. Open `pentaho/server/biserver-ee/pentaho-solutions/system/hibernate/hibernate-settings.xml` in a text editor. By default, system indicates the location of the PostgreSQL hibernate configuration file.

```
<config-file>system/hibernate/postgresql.hibernate.cfg.xml</config-file>
```

2. Change the default reference to the MySQL configuration file.

```
<config-file>system/hibernate/mysql5.hibernate.cfg.xml</config-file>
```

3. Save and close the file.

4. Open `pentaho/server/biserver-ee/pentaho-solutions/system/hibernate/mysql5.hibernate.cfg.xml` in a text editor.
5. Make sure that the password and port number match the ones you specified in your configuration. Make changes as necessary, then save and close the file.

Replace Default Version of Audit Log File with MySQL Version

The default audit_sql.xml file that is in the pentaho-solutions/system directory is configured for the PostgreSQL database. Since you are using MySQL to host the BA Repository, you need to replace the `audit_sql.xml` file with one that is configured for MySQL. To do this, copy the `pentaho-solutions/system/dialects/mysql5/audit_sql.xml` file to the `pentaho-solutions/system` directory.

Modify Jackrabbit Repository Information for MySQL

You must indicate which database is used as the BA Repository as well as the port, url, username, and password. All of the information needed to configure the repository for the PostgreSQL, MySQL, and Oracle BA Repository databases appear. By default, the PostgreSQL sections are not commented out, but the MySQL and Oracle sections are. To modify this file so that it works for your BA Repository, you will need to make sure that the sections that refer to your BA Repository Database are not commented out, and the sections refer to other BA Repository databases are commented out.

When code is commented out, it appears between the `<!--` and `-->` tags. The information in between the tags is commented out, and is therefore not executed by the software. In this example, the code `<!` `<FileSystem class="org.apache.jackrabbit.core.fs.db.OracleFileSystem">` is commented out.

```
<!--
<FileSystem class="org.apache.jackrabbit.core.fs.db.OracleFileSystem">
-->
```

To modify the Jackrabbit repository so that MySQL is the default database, do this.

1. Use a text editor to open the `pentaho/server/biserver-ee/pentaho-solutions/system/jackrabbit/repository.xml` file.
2. In the Repository part of the code, change the code so that the MySQL lines of code are not commented out, but the PostgreSQL and Oracle lines are, like this.

```
<FileSystem class="org.apache.jackrabbit.core.fs.db.DbFileSystem">
    <param name="driver" value="com.mysql.jdbc.Driver"/>
    <param name="url" value="jdbc:mysql://localhost:3306/jackrabbit"/>
    <param name="user" value="jcr_user"/>
    <param name="password" value="password"/>
    <param name="schema" value="mysql"/>
    <param name="schemaObjectPrefix" value="fs_repos_"/>
  </FileSystem>
```

```
<!--
<FileSystem class="org.apache.jackrabbit.core.fs.db.OracleFileSystem">
    <param name="url" value="jdbc:oracle:thin:@localhost:1521:orcl"/>
    ...
    <param name="schemaObjectPrefix" value="fs_repos_"/>
    <param name="tablespace" value="jackrabbit"/>
  </FileSystem>

  <FileSystem class="org.apache.jackrabbit.core.fs.db.DbFileSystem">
    <param name="driver" value="org.postgresql.Driver"/>
   ...
    <param name="schemaObjectPrefix" value="fs_repos_"/>
  </FileSystem>
-->
```

Note: If you changed your password when you initialized the database during the Prepare Environment step, or if your database is on a different port, edit the url and password parameters accordingly.

3. In the DataStore section of the code, change the code so that the MySQL lines of code are not commented out, but the PostgreSQL and Oracle lines are, like this.

```
<DataStore class="org.apache.jackrabbit.core.data.db.DbDataStore">
    <param name="url" value="jdbc:mysql://localhost:3306/jackrabbit"/>
    <param name="user" value="jcr_user"/>
    <param name="password" value="password"/>
    <param name="databaseType" value="mysql"/>
    <param name="driver" value="com.mysql.jdbc.Driver"/>
    <param name="minRecordLength" value="1024"/>
    <param name="maxConnections" value="3"/>
    <param name="copyWhenReading" value="true"/>
    <param name="tablePrefix" value=""/>
    <param name="schemaObjectPrefix" value="ds_repos_"/>
  </DataStore>
 <!--
 <DataStore class="org.apache.jackrabbit.core.data.db.DbDataStore">
    <param name="url" value="jdbc:oracle:thin:@localhost:1521:orcl"/>
   ...
    <param name="schemaObjectPrefix" value="ds_repos_"/>
  </DataStore>
  <DataStore class="org.apache.jackrabbit.core.data.db.DbDataStore">
    <param name="url" value="jdbc:postgresql://localhost:5432/jackrabbit"/>
  ...
    <param name="schemaObjectPrefix" value="ds_repos_"/>
```

```
  </DataStore>
  -->
```

4. In the Workspaces section of the code, change the code so that the MySQL lines of code are not commented out, but the PostgreSQL and Oracle lines are, like this.

```
<FileSystem class="org.apache.jackrabbit.core.fs.db.DbFileSystem">
      <param name="driver" value="com.mysql.jdbc.Driver"/>
      <param name="url" value="jdbc:mysql://localhost:3306/jackrabbit"/>
      <param name="user" value="jcr_user"/>
      <param name="password" value="password"/>
      <param name="schema" value="mysql"/>
      <param name="schemaObjectPrefix" value="fs_ws_"/>
    </FileSystem>
<!--
    <FileSystem class="org.apache.jackrabbit.core.fs.db.OracleFileSystem">
      <param name="url" value="jdbc:oracle:thin:@localhost:1521:orcl"/>
     ...
      <param name="schemaObjectPrefix" value="fs_ws_"/>
      <param name="tablespace" value="jcr_user"/>
    </FileSystem>
    <FileSystem class="org.apache.jackrabbit.core.fs.db.DbFileSystem">
      <param name="driver" value="org.postgresql.Driver"/>
   ...
      <param name="schemaObjectPrefix" value="fs_ws_"/>
    </FileSystem>
-->
```

Note: If you changed your password when you initialized the database during the Prepare Environment step, or if your database is on a different port, edit the url and password parameters accordingly.

5. In the Persistence Manager section of the code, change the code so that the MySQL lines of code are not commented out, but the PostgreSQL and Oracle lines are, like this.

```
<PersistenceManager class="org.apache.jackrabbit.core.persistence.bundle.
MySqlPersistenceManager">
      <param name="url" value="jdbc:mysql://localhost:3306/jackrabbit"/>
      <param name="user" value="jcr_user" />
      <param name="password" value="password" />
      <param name="schema" value="mysql"/>
      <param name="schemaObjectPrefix" value="${wsp.name}_pm_ws_"/>
    </PersistenceManager>
 <!--
    <PersistenceManager class="org.apache.jackrabbit.core.persistence.
```

```
bundle.OraclePersistenceManager">
      <param name="url" value="jdbc:oracle:thin:@localhost:1521:orcl"/>
     ...
      <param name="schemaObjectPrefix" value="${wsp.name}_pm_ws_"/>
      <param name="tablespace" value="jackrabbit"/>
    </PersistenceManager>
    <PersistenceManager class="org.apache.jackrabbit.core.persistence.
bundle.PostgreSQLPersistenceManager">
      <param name="url" value="jdbc:postgresql://localhost:5432/
jackrabbit"/>
     ...
      <param name="schemaObjectPrefix" value="${wsp.name}_pm_ws_"/>
    </PersistenceManager>
 -->
```

Note: If you changed your password when you initialized the database during the Prepare Environment step, or if your database is on a different port, edit the url and password parameters accordingly.

6. In the Versioning section of the code, change the code so that the MySQL lines of code are not commented out, but the PostgreSQL and Oracle lines are, like his.

```
<FileSystem class="org.apache.jackrabbit.core.fs.db.DbFileSystem">
      <param name="driver" value="com.mysql.jdbc.Driver"/>
      <param name="url" value="jdbc:mysql://localhost:3306/jackrabbit"/>
      <param name="user" value="jcr_user"/>
      <param name="password" value="password"/>
      <param name="schema" value="mysql"/>
      <param name="schemaObjectPrefix" value="fs_ver_"/>
    </FileSystem>
<!--
    <FileSystem class="org.apache.jackrabbit.core.fs.db.OracleFileSystem">
      <param name="url" value="jdbc:oracle:thin:@localhost:1521:orcl"/>
    ...
      <param name="schemaObjectPrefix" value="fs_ver_"/>
      <param name="tablespace" value="jackrabbit"/>
    </FileSystem>
    <FileSystem class="org.apache.jackrabbit.core.fs.db.DbFileSystem">
      <param name="driver" value="org.postgresql.Driver"/>
      ...
      <param name="schemaObjectPrefix" value="fs_ver_"/>
    </FileSystem>
-->
```

Note: If you changed your password when you initialized the database during the Prepare Environment step, or if your database is on a different port, edit the url and password parameters accordingly.

7. In the Persistence Manager section of the code that is near the end of the file, change the code so that the MySQL lines of code are not commented out, but the PostgreSQL and Oracle lines are, like this.

```
<PersistenceManager class="org.apache.jackrabbit.core.persistence.bundle.
MySqlPersistenceManager">
      <param name="url" value="jdbc:mysql://localhost:3306/jackrabbit"/>
      <param name="user" value="jcr_user" />
      <param name="password" value="password" />
      <param name="schema" value="mysql"/>
      <param name="schemaObjectPrefix" value="pm_ver_"/>
    </PersistenceManager>
<!--
    <PersistenceManager class="org.apache.jackrabbit.core.persistence.
bundle.OraclePersistenceManager">
      <param name="url" value="jdbc:oracle:thin:@localhost:1521:orcl"/>
     ...
      <param name="schemaObjectPrefix" value="pm_ver_"/>
      <param name="tablespace" value="jackrabbit"/>
    </PersistenceManager>
    <PersistenceManager class="org.apache.jackrabbit.core.persistence.
bundle.PostgreSQLPersistenceManager">
      <param name="url" value="jdbc:postgresql://localhost:5432/
jackrabbit"/>
     ...
      <param name="schemaObjectPrefix" value="pm_ver_"/>
    </PersistenceManager>
-->
```

Note: If you changed your password when you initialized the database during the Prepare Environment step, or if your database is on a different port, edit the url and password parameters accordingly.

Prepare Oracle BA Repository Database

These instructions explain how to configure Quartz, Hibernate, Jackrabbit, and Pentaho Security. By default, the examples in this section are for a Oracle database that runs on port 1521. The default password is also in these examples. If you have a different port, different password complete all of the instructions in these steps.

Configure Quartz on Oracle BA Repository Database

When you use Pentaho to schedule an event, such as a report to be run every Sunday at 1:00 a.m. EST, event information is stored in the Quartz JobStore. During the installation process, you must indicate where the JobStore is located. To do this, modify the `quartz.properties` file.

1. Open the `pentaho/server/biserver-ee/pentaho-solutions/system/quartz/quartz.properties` file in the text editor of your choice.
2. In the `#_replace_jobstore_properties` section of the file, set the `org.quartz.jobStore.driverDelegateClass` equal to `org.quartz.impl.jdbcjobstore.oracle.OracleDelegate`.

```
org.quartz.jobStore.driverDelegateClass = org.quartz.impl.jdbcjobstore.
oracle.OracleDelegate
```

3. Save the file and close the text editor.

Configure Hibernate Settings for Oracle

Modify the hibernate settings file to specify where Pentaho will find the BA Repository's hibernate configuration file. The hibernate configuration file specifies driver and connection information, as well as dialects and how to handle connection closes and timeouts.

1. Open `pentaho/biserver-ee/pentaho-solutions/system/hibernate/hibernate-settings.xml` in a text editor. By default, system indicates the location of the PostgreSQL hibernate configuration file.

```
<config-file>system/hibernate/postgresql.hibernate.cfg.xml</config-file>
```

2. Change the default to this to point to the Oracle configuration file.

```
<config-file>system/hibernate/oracle10g.hibernate.cfg.xml</config-file>
```

3. Save and close the file.
4. Open `pentaho/biserver-ee/system/hibernate/oracle10g.hibernate.cfg.xml` in a text editor.
5. Make sure that the password and port number match the ones you specified in your configuration. Make changes as necessary, then save and close the file.

Replace Default Version of Audit Log File with Oracle Version

The default audit_sql.xml file that is in the pentaho-solutions/system directory is configured for the PostgreSQL database. Since you are using Oracle to host the BA Repository, you need to replace the `audit_sql.xml` file with one that is configured for Oracle. To do this, copy the `pentaho-solutions/system/dialects/oracle10g/audit_sql.xml` file to the `pentaho-solutions/system` directory.

Modify Jackrabbit Repository Information for Oracle

You must indicate which database is used as the BA Repository as well as the port, url, username, and password. All of the information needed to configure the repository for the PostgreSQL, MySQL, and Oracle BA Repository databases appear. By default, the PostgreSQL sections are not commented out, but the MySQL and Oracle sections are. To modify this file so that it works for your BA Repository, you will need to make sure that the sections that refer to your BA Repository Database are not commented out, and the sections refer to other BA Repository databases are commented out.

When code is commented out, it appears between the <!-- and --> tags. The information in between the tags is commented out, and is therefore not executed by the software. In this example, the code <!
<FileSystem class="org.apache.jackrabbit.core.fs.db.OracleFileSystem"> is commented out.

```
<!--
<FileSystem class="org.apache.jackrabbit.core.fs.db.OracleFileSystem">
-->
```

To modify the Jackrabbit repository so that Oracle is the default database, do this.

1. Use a text editor to open the pentaho/server/biserver-ee/pentaho-solutions/system/jackrabbit/repository.xml file.
2. In the Repository part of the code, change the code so that the Oracle lines of code are not commented out, but the PostgreSQL and MySQL lines are, like this.

```
<!--
<FileSystem class="org.apache.jackrabbit.core.fs.db.DbFileSystem">
    <param name="driver" value="com.mysql.jdbc.Driver"/>
    <param name="url" value="jdbc:mysql://localhost:3306/jackrabbit"/>
   ...
    <param name="schemaObjectPrefix" value="fs_repos_"/>
  </FileSystem>
-->
<FileSystem class="org.apache.jackrabbit.core.fs.db.OracleFileSystem">
    <param name="url" value="jdbc:oracle:thin:@localhost:1521:orcl"/>
    <param name="user" value="jcr_user"/>
    <param name="password" value="password"/>
    <param name="schemaObjectPrefix" value="fs_repos_"/>
    <param name="tablespace" value="jackrabbit"/>
  </FileSystem>
<!--
  <FileSystem class="org.apache.jackrabbit.core.fs.db.DbFileSystem">
    <param name="driver" value="org.postgresql.Driver"/>
    <param name="url" value="jdbc:postgresql://localhost:5432/jackrabbit"/>
    ...
    <param name="schemaObjectPrefix" value="fs_repos_"/>
  </FileSystem>
-->
```

Note: If you changed your password when you initialized the database during the Prepare Environment step, or if your database is on a different port, edit the url and password parameters accordingly.

3. Change <param name="tablespace" value="jackrabbit"/> to <param name="tablespace" value="pentaho_tablespace"/>.

4. In the DataStore section of the code, change the code so that the Oracle lines of code are not commented out, but the PostgreSQL and MySQL lines are, like this.

```
<!--
<DataStore class="org.apache.jackrabbit.core.data.db.DbDataStore">
    <param name="url" value="jdbc:mysql://localhost:3306/jackrabbit"/>
    ...
    <param name="schemaObjectPrefix" value="ds_repos_"/>
  </DataStore>
  -->
 <DataStore class="org.apache.jackrabbit.core.data.db.DbDataStore">
    <param name="url" value="jdbc:oracle:thin:@localhost:1521:orcl"/>
    <param name="driver" value="oracle.jdbc.driver.OracleDriver"/>
    <param name="user" value="jcr_user"/>
    <param name="password" value="password"/>
    <param name="databaseType" value="oracle"/>
    <param name="minRecordLength" value="1024"/>
    <param name="maxConnections" value="3"/>
    <param name="copyWhenReading" value="true"/>
    <param name="tablePrefix" value=""/>
    <param name="schemaObjectPrefix" value="ds_repos_"/>
  </DataStore>
 <!--
 <DataStore class="org.apache.jackrabbit.core.data.db.DbDataStore">
    <param name="url" value="jdbc:postgresql://localhost:5432/jackrabbit"/>
    ...
    <param name="schemaObjectPrefix" value="ds_repos_"/>
  </DataStore>
```

Note: If you changed your password when you initialized the database during the Prepare Environment step, or if your database is on a different port, edit the url and password parameters accordingly.

5. In the Workspaces section of the code, change the code so that the Oracle lines of code are not commented out, but the PostgreSQL and MySQL lines are, like this.

```
<!--
<FileSystem class="org.apache.jackrabbit.core.fs.db.DbFileSystem">
      <param name="driver" value="com.mysql.jdbc.Driver"/>
      <param name="url" value="jdbc:mysql://localhost:3306/jackrabbit"/>
     ...
      <param name="schemaObjectPrefix" value="fs_ws_"/>
    </FileSystem>
-->
    <FileSystem class="org.apache.jackrabbit.core.fs.db.OracleFileSystem">
```

```
      <param name="url" value="jdbc:oracle:thin:@localhost:1521:orcl"/>
      <param name="user" value="jcr_user"/>
      <param name="password" value="password"/>
      <param name="schemaObjectPrefix" value="fs_ws_"/>
      <param name="tablespace" value="jcr_user"/>
    </FileSystem>
<!--
    <FileSystem class="org.apache.jackrabbit.core.fs.db.DbFileSystem">
      <param name="driver" value="org.postgresql.Driver"/>
      <param name="url" value="jdbc:postgresql://localhost:5432/
jackrabbit"/>
      ...
      <param name="schemaObjectPrefix" value="fs_ws_"/>
    </FileSystem>
-->
```

Note: If you changed your password when you initialized the database during the Prepare Environment step, or if your database is on a different port, edit the url and password parameters accordingly.

6. Change <param name="tablespace" value="jcr_user"/> to <param name="tablespace" value="pentaho_tablespace"/>.
7. In the Persistence Manager section of the code, change the code so that the Oracle lines of code are not commented out, but the PostgreSQL and MySQL lines are, like this.

```
<!--
<PersistenceManager class="org.apache.jackrabbit.core.persistence.bundle.
MySqlPersistenceManager">
      <param name="url" value="jdbc:mysql://localhost:3306/jackrabbit"/>
     ...
      <param name="schemaObjectPrefix" value="${wsp.name}_pm_ws_"/>
    </PersistenceManager>
 -->
    <PersistenceManager class="org.apache.jackrabbit.core.persistence.
bundle.OraclePersistenceManager">
      <param name="url" value="jdbc:oracle:thin:@localhost:1521:orcl"/>
      <param name="driver" value="oracle.jdbc.driver.OracleDriver"/>
      <param name="user" value="jcr_user"/>
      <param name="password" value="password"/>
      <param name="schema" value="oracle"/>
      <param name="schemaObjectPrefix" value="${wsp.name}_pm_ws_"/>
      <param name="tablespace" value="jackrabbit"/>
    </PersistenceManager>
 <!--
```

```
   <PersistenceManager class="org.apache.jackrabbit.core.persistence.bundle.
PostgreSQLPersistenceManager">
      <param name="url" value="jdbc:postgresql://localhost:5432/
jackrabbit"/>
      ...
      <param name="schemaObjectPrefix" value="${wsp.name}_pm_ws_"/>
    </PersistenceManager>
 -->
```

Note: If you changed your password when you initialized the database during the Prepare Environment step, or if your database is on a different port, edit the url and password parameters accordingly.

8. Change <param name="tablespace" value="jackrabbit"/> to <param name="tablespace" value="pentaho_tablespace"/>.
9. In the Versioning section of the code, change the code so that the Oracle lines of code are not commented out, but the PostgreSQL and MySQL lines are, like his.

```
<!--
<FileSystem class="org.apache.jackrabbit.core.fs.db.DbFileSystem">
      <param name="driver" value="com.mysql.jdbc.Driver"/>
      <param name="url" value="jdbc:mysql://localhost:3306/jackrabbit"/>
      ...
      <param name="schemaObjectPrefix" value="fs_ver_"/>
    </FileSystem>
-->
    <FileSystem class="org.apache.jackrabbit.core.fs.db.OracleFileSystem">
      <param name="url" value="jdbc:oracle:thin:@localhost:1521:orcl"/>
      <param name="user" value="jcr_user"/>
      <param name="password" value="password"/>
      <param name="schemaObjectPrefix" value="fs_ver_"/>
      <param name="tablespace" value="jackrabbit"/>
    </FileSystem>
 <!--
   <FileSystem class="org.apache.jackrabbit.core.fs.db.DbFileSystem">
      <param name="driver" value="org.postgresql.Driver"/>
      <param name="url" value="jdbc:postgresql://localhost:5432/
jackrabbit"/>
      ...
      <param name="schemaObjectPrefix" value="fs_ver_"/>
    </FileSystem>
-->
```

Note: If you changed your password when you initialized the database during the Prepare Environment step, or if your database is on a different port, edit the url and password parameters accordingly.

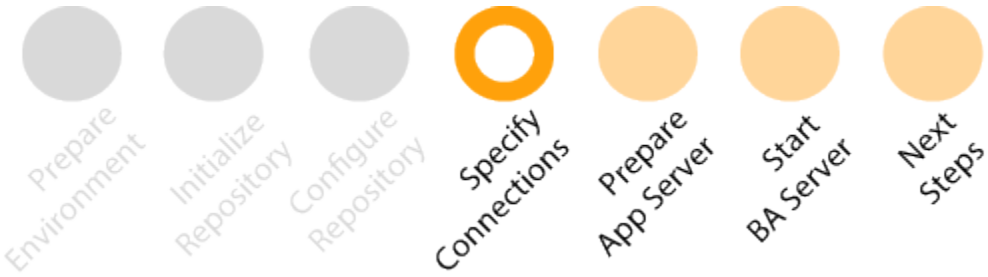
10. Change <param name="tablespace" value="jackrabbit"/> to <param name="tablespace" value="pentaho_tablespace"/>.
11. In the Persistence Manager section of the code that is near the end of the file, change the code so that the Oracle lines of code are not commented out, but the PostgreSQL and MySQL lines are, like this.

```
<!--
<PersistenceManager class="org.apache.jackrabbit.core.persistence.bundle.
MySqlPersistenceManager">
      <param name="url" value="jdbc:mysql://localhost:3306/jackrabbit"/>
      ...
      <param name="schemaObjectPrefix" value="pm_ver_"/>
    </PersistenceManager>
-->
    <PersistenceManager class="org.apache.jackrabbit.core.persistence.
bundle.OraclePersistenceManager">
      <param name="url" value="jdbc:oracle:thin:@localhost:1521:orcl"/>
      <param name="driver" value="oracle.jdbc.driver.OracleDriver"/>
      <param name="user" value="jcr_user"/>
      <param name="password" value="password"/>
      <param name="schema" value="oracle"/>
      <param name="schemaObjectPrefix" value="pm_ver_"/>
      <param name="tablespace" value="jackrabbit"/>
    </PersistenceManager>
 <!--
   <PersistenceManager class="org.apache.jackrabbit.core.persistence.bundle.
PostgreSQLPersistenceManager">
      <param name="url" value="jdbc:postgresql://localhost:5432/
jackrabbit"/>
     ...
      <param name="schemaObjectPrefix" value="pm_ver_"/>
    </PersistenceManager>
-->
```

Note: If you changed your password when you initialized the database during the Prepare Environment step, or if your database is on a different port, edit the url and password parameters accordingly.

12. Change <param name="tablespace" value="jackrabbit"/> to <param name="tablespace" value="pentaho_tablespace"/>.

Specify Connections



After your repository has been configured, you must configure the web application servers to connect to the BA Repository. In this step, JDBC and JNDI connections are made to the Hibernate, Jackrabbit, and Quartz databases. These databases were installed on your BA Repository database during the Initialize Repository and Configure Repository sections of these instructions.

By default, the BA Server software is configured to be deployed and run on the Tomcat server. As such, connections have already been specified and only the Tomcat context.xml file must be modified. For JBoss, both JDBC and JNDI connection information must be specified. Since what must be completed varies according to web server, tasks in this section are grouped according to the web application server you have installed.

If you have Tomcat, complete the following tasks.

- Download and Install Repository Database JDBC Drivers
- Download and Install H2 Driver
- Modify JDBC Connection Information

If you have JBoss, complete the following tasks.

- Define JNDI Connection Information
- Remove JNDI Resource References
- Update JNDI Data Source References to Conform with JBoss Standards

Perform Tomcat-Specific Connection Tasks

If you plan to run the BA Server on Tomcat, you must modify JDBC Connection information.

Download and Install Repository Database JDBC Drivers

For the BA Server to connect to the BA Repository database of your choice, add the BA Repository's database JDBC driver library to the appropriate place in the web application server on which the BA Server will be deployed. The default web application server for the archive installation process is Tomcat.

1. 1. Download a JDBC driver JAR from your database vendor or a third-party driver developer.

Due to licensing restrictions, Pentaho does not distribute the necessary JDBC driver JARs. This is why you have to download the file yourself and install it.

2. 2. Copy the JDBC driver JAR you just downloaded to the `<your tomcat installation directory>/lib/` directory.

Download and Install H2 JDBC Drivers

Install the H2 JDBC driver.

1. Download the version 1.2.131 of the H2 JDBC JAR from the http://code.google.com/p/h2database/downloads/list.
2. Copy the JDBC driver JAR you just downloaded to the `<your tomcat installation directory>/lib/` directory.

Modify JDBC Connection Information in the Tomcat context.xml File

Database connection and network information, such as the username, password, driver class information, IP address or domain name, and port numbers for your BA Repository database are stored in the `context.xml` file. Modify this file to reflect the database connection and network information to reflect your operating environment. You also modify the values for the `validationQuery` parameters in this file if you have choosen to use an BA Repository database other than PostgreSQL.

1. Consult your database documentation to determine the JDBC class name and connection string for your BA Repository database.
2. View the contents of the `pentaho.war` file with a zip utility, such as 7-Zip, WinZip, or Archive. Do not unzip the file.
3. Use a text editor of your choice to open the `context.xml` file that is in Tomcat's `META-INF/` directory.
4. Do one of these things.
    a. If you are using PostgreSQL as your BA Repository database, comment out the resource references that refer to other databases, such as PostgreSQL and Oracle. In the PostgreSQL sections make sure tha the code looks like the following, but adjust the port numbers and passwords to reflect your environment, if necessary.

```
<Resource validationQuery="select 1"
url="jdbc:postgresql://localhost:5432/hibernate" driverClassName="org.
postgresql.Driver" password="password" username="hibuser"
maxWait="10000" maxIdle="5" maxActive="20" factory="org.apache.
commons.dbcp.BasicDataSourceFactory" type="javax.sql.DataSource"
auth="Container" name="jdbc/Hibernate"/>
<Resource validationQuery="select 1"
url="jdbc:postgresql://localhost:5432/hibernate" driverClassName="org.
postgresql.Driver" password="password" username="hibuser"
maxWait="10000" maxIdle="5" maxActive="20" factory="org.apache.
commons.dbcp.BasicDataSourceFactory" type="javax.sql.DataSource"
auth="Container" name="jdbc/Audit"/>
<Resource validationQuery="select 1"
```

```
url="jdbc:postgresql://localhost:5432/quartz" driverClassName="org.
postgresql.Driver" password="password" username="pentaho_user"
maxWait="10000" maxIdle="5" maxActive="20" factory="org.apache.
commons.dbcp.BasicDataSourceFactory" type="javax.sql.DataSource"
auth="Container" name="jdbc/Quartz"/>
<Resource validationQuery="select 1"
url="jdbc:postgresql://localhost:5432/hibernate" driverClassName="org.
postgresql.Driver" password="password" username="hibuser"
maxWait="10000" maxIdle="5" maxActive="20" factory="org.apache.
commons.dbcp.BasicDataSourceFactory" type="javax.sql.DataSource"
auth="Container" name="jdbc/pentaho_operations_mart"/>
<Resource validationQuery="select 1"
url="jdbc:postgresql://localhost:5432/hibernate" driverClassName="org.
postgresql.Driver" password="password" username="hibuser"
maxWait="10000" maxIdle="5" maxActive="20" factory="org.apache.
commons.dbcp.BasicDataSourceFactory" type="javax.sql.DataSource"
auth="Container" name="jdbc/PDI_Operations_Mart"/>
```

b. If you are using MySQL as your BA Repository database, comment out the resource references that refer to other databases, such as PostgreSQL and Oracle. Then, add the following code to the file if it does not already exist. Adjust the port numbers and passwords to reflect your environment, if necessary.

```
<Resource validationQuery="select 1" url="jdbc:mysql://localhost:3306/
hibernate" driverClassName="com.mysql.jdbc.Driver"
password="password" username="hibuser" maxWait="10000" maxIdle="5"
maxActive="20" factory="org.apache.commons.dbcp.
BasicDataSourceFactory" type="javax.sql.DataSource" auth="Container"
name="jdbc/Hibernate"/>
<Resource validationQuery="select 1" url="jdbc:mysql://localhost:3306/
hibernate" driverClassName="com.mysql.jdbc.Driver"
password="password" username="hibuser" maxWait="10000" maxIdle="5"
maxActive="20" factory="org.apache.commons.dbcp.
BasicDataSourceFactory" type="javax.sql.DataSource" auth="Container"
name="jdbc/Audit"/>
<Resource validationQuery="select 1" url="jdbc:mysql://localhost:3306/
quartz" driverClassName="com.mysql.jdbc.Driver" password="password"
username="pentaho_user" maxWait="10000" maxIdle="5" maxActive="20"
factory="org.apache.commons.dbcp.BasicDataSourceFactory" type="javax.
sql.DataSource" auth="Container" name="jdbc/Quartz"/>
<Resource validationQuery="select 1" url="jdbc:mysql://localhost:3306/
pentaho_operations_mart" driverClassName="com.mysql.jdbc.Driver"
```

```
password="password" username="hibuser" maxWait="10000" maxIdle="5"
maxActive="20" factory="org.apache.commons.dbcp.
BasicDataSourceFactory" type="javax.sql.DataSource" auth="Container"
name="jdbc/pentaho_operations_mart"/>
<Resource validationQuery="select 1" url="jdbc:mysql://localhost:3306/
pentaho_operations_mart" driverClassName="com.mysql.jdbc.Driver"
password="password" username="hibuser" maxWait="10000" maxIdle="5"
maxActive="20" factory="org.apache.commons.dbcp.
BasicDataSourceFactory" type="javax.sql.DataSource" auth="Container"
name="jdbc/PDI_Operations_Mart"/>
```

c. If you are using Oracle as your BA Repository database, comment out the resource references that refer to other databases such as PostgreSQL and MySQL. Then, add the following code to the file if it does not exist. Adjust the port numbers and passwords to reflect your environment, if necessary.

```
<Resource validationQuery="select 1 from dual"
url="jdbc:oracle:thin:@localhost:1521/XE" driverClassName="oracle.
jdbc.OracleDriver" password="password" username="hibuser"
maxWait="10000" maxIdle="5" maxActive="20" factory="org.apache.
commons.dbcp.BasicDataSourceFactory" type="javax.sql.DataSource"
auth="Container" name="jdbc/Hibernate"/>
<Resource validationQuery="select 1 from dual"
url="jdbc:oracle:thin:@localhost:1521/XE" driverClassName="oracle.
jdbc.OracleDriver" password="password" username="hibuser"
maxWait="10000" maxIdle="5" maxActive="20" factory="org.apache.
commons.dbcp.BasicDataSourceFactory" type="javax.sql.DataSource"
auth="Container" name="jdbc/Audit"/>
<Resource validationQuery="select 1 from dual"
url="jdbc:oracle:thin:@localhost:1521/XE" driverClassName="oracle.
jdbc.OracleDriver" password="password" username="quartz"
maxWait="10000" maxIdle="5" maxActive="20" factory="org.apache.
commons.dbcp.BasicDataSourceFactory" type="javax.sql.DataSource"
auth="Container" name="jdbc/Quartz"/>
<Resource validationQuery="select 1 from dual"
url="jdbc:oracle:thin:@localhost:1521/XE" driverClassName="oracle.
jdbc.OracleDriver" password="password" username="hibuser"
maxWait="10000" maxIdle="5" maxActive="20" factory="org.apache.
commons.dbcp.BasicDataSourceFactory" type="javax.sql.DataSource"
auth="Container" name="jdbc/pentaho_operations_mart"/>
<Resource validationQuery="select 1 from dual"
url="jdbc:oracle:thin:@localhost:1521/XE" driverClassName="oracle.
```

```
jdbc.OracleDriver" password="password" username="hibuser"
maxWait="10000" maxIdle="5" maxActive="20" factory="org.apache.
commons.dbcp.BasicDataSourceFactory" type="javax.sql.DataSource"
auth="Container" name="jdbc/PDI_Operations_Mart"/>
```

5. Modify the username, password, driver class information, IP address (or domain name), and port numbers so they reflect the correct values for your environment.
6. Make sure that the `validationQuery` variable for your database is set to one of these.
   - **PostgreSQL**`validationQuery="select 1"`
   - **MySQL:**`validationQuery="/* ping */ select 1"/`
   - **Oracle:**`validationQuery="select 1 from dual"`

10. Save the `context.xml`file, then close it.
11. To verify that the changes have been made and saved, do this.
    a. View the `context.xml` file. Make sure that the changes that you made to the file in these instructions have been saved.
    b. Close the file.
12. To make sure that the changes that you made in the context.xml file take effect when Tomcat is started, navigate to the `<your tomcat installation directory>\conf\Catalina\localhost` directory. If the `pentaho.xml` file is in the present, delete it. It will be generated again when you start the BA Server, but will contain the changes that you just made in the `context.xml` file.

Perform JBoss-Specific Connection Tasks

To define JDBC and JNDI connections, several JBoss-specific tasks are required.

Install JDBC Driver as a Module in JBoss

In JBoss, JDBC driver information is stored in a module, which is an XML file that you create. You must download the JDBC driver software component to the correct directory, then create `module.xml` files for each database. You need to create a file for the database that hosts the BA Repository (either PostgreSQL, MySQL, or Oracle), as well as for HSQLDB and H2.

1. 1. Navigate to the `pentaho/server/biserver-ee/<your jboss installation directory>/modules/system/layers/base/org` folder and create one of the following paths for the database on which you are hosting the BA Repository.
   - • **PostgreSQL:** `postgresql/main`
   - • **MySQL:** `mysql/main`
   - • **Oracle:** `oracle/main`

5. 2. Navigate to the pentaho/server/biserver-ee/<your jboss installation directory>/modules/system/layers/base/com/h2database directory and create the following two paths.
   - **HSQLDB:** hsqldb`/main`
   - **H2:** h2`/main`

0. 1. Download the supported JDBC driver for your BA Repository database to the `postgresql/main`, `mysql/main`, or `oracle/main` directories (which ever one you created). See the JDBC Drivers Reference for a list of supported drivers.

1. 2. Download the supported JDBC drivers for HSQLDB and H2. Place HQSLDB's driver in the hsqldb/main directory . Place H2's driver in the h2/main directory. See the JDBC Drivers Reference for a list of supported drivers.
2. 3. In the `postgresql/main`, `mysql/main`, or `oracle/main` (which ever one you created), do the following things.
   c. A. Use an editor to create a text file named `module.xml`.
   d. B. Copy the following code to the `module.xml` file, then modify it so that the name of the JDBC driver you just downloaded appears.

```
<?xml version="1.0" encoding="UTF-8"?>
    <module xmlns="urn:jboss:module:1.0" name="org.postgresql">
        <resources>
            <resource-root path="[Name of JDBC Jar You Downloaded Here]"/>
        </resources>
        <dependencies><module name="javax.api"/></dependencies>
    </module>
    <?xml version="1.0" encoding="UTF-8"?>
        <module xmlns="urn:jboss:module:1.0" name="org.mysql">
           <resources>
              <resource-root path="[Name of JDBC Jar You Downloaded
Here]"/>
              </resources>
             <dependencies><module name="javax.api"/></dependencies>
     </module>
```

   - • **PostgreSQL:** If you are using PostgreSQL, copy the following code in the `module.xml` file. Replace the name of the `resource-root path` parameter with the name of the JDBC driver you downloaded.
   - • **MySQL:** If you are using MySQL, copy the following code in the `module.xml` file. Replace the name of the `resource-root path` parameter with the name of the JDBC driver you downloaded.
   - • **Oracle:** If you are using Oracle, copy the following code in the `module.xml` file. Replace the name of the `resource-root path` parameter with the name of the JDBC driver you downloaded.

```
<?xml version="1.0" encoding="UTF-8"?>
    <module xmlns="urn:jboss:module:1.0" name="org.oracle">
        <resources>
            <resource-root path="[Name of JDBC Jar You Downloaded Here]"/>
        </resources>
        <dependencies><module name="javax.api"/></dependencies>
    </module>
```

   h. C. Save and close the `module.xml` file.
9. 4. In the `hsqldb/main` directory, do these things.
   i. A. Use an editor to create a text file named `module.xml`.
   j. B. Copy the following code to the module.xml file, then modify it so that the name of the JDBC driver you just downloaded appears in the `resource-root` path.

```
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.0" name="com.h2database.hsqldb">
<resources>
<resource-root path="[Name of JDBC Jar You Downloaded Here]"/>
</resources>
<dependencies><module name="javax.api"/></dependencies>
</module>
```

10. 1. Save and close the `module.xml` file.
11. 2. In the `h2/main` directory, do these things.
    k. A. Use an editor to create a text file named `module.xml`.
    l. B. Copy the following code to the `module.xml` file, then modify it so that the name of the JDBC driver you just downloaded appears in the `resource-root` path.

```
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.0" name="com.h2database.h2">
<resources>
<resource-root path="[Name of JDBC Jar You Downloaded Here]"/>
</resources>
<dependencies><module name="javax.api"/></dependencies>
</module>
```

12. 1. Save and close the `module.xml` file.

Define JNDI Database Connection Information in JBoss

JNDI is used to specify port, driver, user name, and password information for the Audit and Quartz databases that are housed on your BA Repository database. JNDI provides a common interface for different naming services, such as DNS, LDAP, and Microsoft Active Directory. Instead of having to remember the details for how to connect or interact with the data for many different naming services, you need to only use an XML to specify the information you want to pass to the naming service.

1. Use a text editor to open the `pentaho/server/biserver-ee/<your jboss installation directory>/standalone/configuration/standalone.xml` file.
2. Insert these lines after the definition of `ExampleDS` data source.

```
<datasource jndi-name="java:jboss/datasources/Hibernate" pool-
name="hibpool" enabled="true" jta="true" use-java-context="true" use-
ccm="true">
                        <connection-url>
                            jdbc:postgresql://localhost:5432/hibernate
                        </connection-url>
                        <driver-class>
```

```
                            org.postgresql.Driver
                        </driver-class>
                        <driver>
                            org.postgresql
                        </driver>
                        <pool>
                            <prefill>
                                false
                            </prefill>
                            <use-strict-min>
                                false
                            </use-strict-min>
                            <flush-strategy>
                                FailingConnectionOnly
                            </flush-strategy>
                        </pool>
                        <security>
                            <user-name>
                                hibuser
                            </user-name>
                            <password>
                                password
                            </password>
                        </security>
                    </datasource>
                    <datasource jndi-name="java:jboss/datasources/Quartz"
pool-name="quartzpool" enabled="true" jta="true" use-java-context="true"
use-ccm="true">
                        <connection-url>
                            jdbc:postgresql://localhost:5432/quartz
                        </connection-url>
                        <driver-class>
                            org.postgresql.Driver
                        </driver-class>
                        <driver>
                            org.postgresql
                        </driver>
                        <pool>
                            <prefill>
                                false
```

```
                            </prefill>
                            <use-strict-min>
                                false
                            </use-strict-min>
                            <flush-strategy>
                                FailingConnectionOnly
                            </flush-strategy>
                        </pool>
                        <security>
                            <user-name>
                                pentaho_user
                            </user-name>
                            <password>
                                password
                            </password>
                        </security>
                    </datasource>
                    <datasource jndi-name="java:jboss/datasources/Audit"
pool-name="auditpool" enabled="true" jta="true" use-java-context="true" use-
ccm="true">
                        <connection-url>
                            jdbc:postgresql://localhost:5432/hibernate
                        </connection-url>
                        <driver-class>
                            org.postgresql.Driver
                        </driver-class>
                        <driver>
                            org.postgresql
                        </driver>
                        <pool>
                            <prefill>
                                false
                            </prefill>
                            <use-strict-min>
                                false
                            </use-strict-min>
                            <flush-strategy>
                                FailingConnectionOnly
                            </flush-strategy>
                        </pool>
```

```
                        <security>
                            <user-name>
                                pentaho_user
                            </user-name>
                            <password>
                                password
                            </password>
                        </security>
                    </datasource>
 <datasource jndi-name="java:jboss/datasources/Pentaho_Operations_Mart"
pool-name="Pentaho_Operations_Mart" enabled="true" jta="true" use-java-
context="true" use-ccm="true">
     <connection-url>
         jdbc:postgresql://localhost:5432/hibernate
     </connection-url>
     <driver-class>
         org.postgresql.Driver
     </driver-class>
     <driver>
         org.postgresql
     </driver>
     <pool>
         <prefill>
             false
         </prefill>
         <use-strict-min>
             false
         </use-strict-min>
         <flush-strategy>
             FailingConnectionOnly
         </flush-strategy>
     </pool>
     <security>
         <user-name>
             hiuser
         </user-name>
         <password>
            password
         </password>
```

```
    </security>
</datasource>
```

3. If your environment (e.g. port numbers, IP address), solution repository, or database password and username information differs from the code you added in the previous step, modify it to match your specifications.
4. Add the driver definition in the driver section of the file. Here is an example of the PostgreSQL driver definition. If you are using MySQL or Oracle, modify the driver name, module, and data source class accordingly.

```
<driver name="org.postgresql" module="org.postgresql">
  <xa-datasource-class>
      org.postgresql.xa.PGXADataSource
      </xa-datasource-class>
  </driver>
```

5. Close and save the `standalone.xml`file.
6. Open the `pentaho/server/biserver-ee/pentaho-solutions/system/applicationContext-spring-security-jdbc.xml` file. Change the port number, driver class name, user name, and password to reflect your environment's settings, if necessary. When complete, save and close the file.

Add JBoss Deployment Structure File to pentaho.war

The `jboss-deployment-structure.xml` file controls class loading. It prevents automatic dependencies from being added, adds dependencies, defines additional modules, changes isolated class loading behavior, and adds additional resource roots to a module. You will need to create, then add a JBoss deployment structure file (`jboss-deploymentstructure.xml`) to the pentaho.war.

1. 1. Use a text editor to create a new file named `jboss-deployment-structure.xml`.
2. 2. Copy the following code snippet to the `jboss-deployment-structure.xml` file.

```
<jboss-deployment-structure>
<deployment>
<exclude-subsystems>
<subsystem name="resteasy" />
<subsystem name="jaxrs" />
<subsystem name="webservices" />
</exclude-subsystems>
<dependencies>
<module name="com.h2database.h2" />
<module name="org.postgresql" />
<module name="com.h2database.hsqldb" />
</dependencies>
</deployment>
</jboss-deployment-structure>
```

0. 1. Save and close the file.
1. 2. Use a zip extraction utility (such as 7-Zip, Winzip, or Archive) to view the contents of the `pentaho.war` file. Do not unzip or extract the contents of the file.
2. 3. Navigate to the `WEB-INF` directory and add the `jboss-deployment-structure.xml` file that you just created to it.
3. 4. Close the `pentaho.war` file. The zip extraction utility that you used might show a prompt that asks whether you would like to update the file in the pentaho.war archive. If this happens, confirm that you would like to do this.

Remove JNDI Resource References in JBoss

Because JBoss has its own mechanism for referencing JNDI data sources, the resource-references in the `web.xml` file located in the `pentaho.war` are not needed. You must remove these resource-references for the BA Server to operate properly.

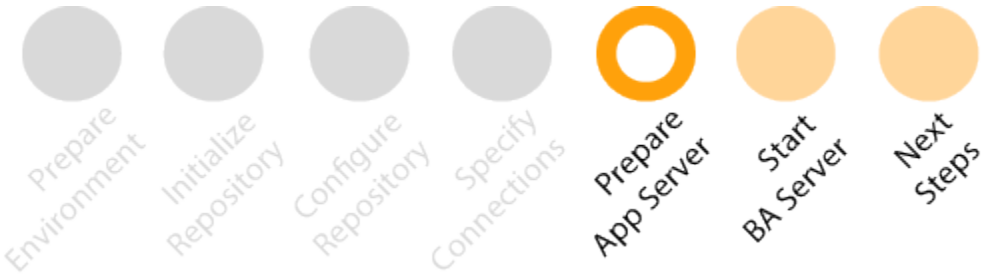
1. Navigate to the `pentaho/server/biserver-ee/<your jboss installation directory>/standalone/deployments` directory.
2. Use a zip extraction utility (such as 7-Zip, Winzip, or Archive) to view the contents of the `pentaho.war` file. Do not unzip or extract the contents of the file.
3. Navigate to the `WEB-INF` directory and open the `web.xml` file in a text editor.
4. Delete all `<resource-ref>` tagged entries including everything between the `<resource-ref>` and `</resource ref>` tags.
5. Save and close the file.
6. The zip extraction utility that you used might show a prompt that asks whether you would like to update the file in the **pentaho.war** archive. If this happens, confirm that you would like to do this.

Update JNDI Data Source Reference to Conform to JBoss Standards

Update these files so that referenced JNDI datasources conform to JBoss standards.

1. Use a text editor to open the `pentaho/server/biserver-ee/pentaho-solutions/system/quartz/quartz.properties` file.
2. Change the `org.quartz.dataSource.myDS.jndiURL` value to `jboss/datasources/Quartz`, then save and close the file.
3. Use a text editor to open the `pentaho/server/biserver-ee/pentaho-solutions/system/audit_sql.xml` file.
4. Change the JNDI value to `jboss/datasources/Hibernate`, then save and close the file.
5. Use a text editor to open the `pentaho/server/biserver-ee/pentaho-solutions/system/data-access/settings.xml` file.
6. Change the `data-access-staging-jndi` value to `jboss/datasources/Hibernate`, then save and close the file.
7. Open the `pentaho/server/biserver-ee/pentaho-solutions/system/audit/dialects/h2` folder. Use the text editor to open each file and make the following changes:
    - Change `<database>Audit</database>` to `<database>jboss/datasources/Audit</database>`.
    - Change `<database>Hibernate</database>` to `<database>jboss/datasources/Hibernate</database>`.

Prepare Web Application Servers



After you have completed the tasks in the Specify Connections step, you must determine whether your web application server must be configured before the BA Server is deployed on it.

Tomcat: No additional configuration is required. Proceed to the Start BA Server step.

JBoss: If you have installed the JBoss web application server, you must manually complete several configuration tasks.

Increase the Amount of Time JBoss Allows for BA Server Deployment

By default, JBoss allows up to one minute for a web application to be deployed. If the web application is not deployed within that timeframe, an error occurs.

Because the BA Server deployment requires more than one minute, manually edit the `standalone.xml` file to increase the deployment time.

1. Use a text editor to open the `<your jboss installation directory>/standalone/configuration/standalone.xml` file.
2. Find the `<deployment-scanner>` tag, add the `deployment-timeout` attribute, then set the attribute equal to `120`. Note that if you are installing the BA Server on a VM, you might want to increase the `deployment-timeout` attribute's value to give the BA Server more time to deploy.
   ```
   <deployment-scanner scan-interval="5000" relative-to="jboss.server.base.dir" path="deployments" scan-enabled="true" deployment-timeout="120"/>
   ```
3. Save and close the file.

Disable the JBoss RESTEasy Scan

To load pentaho REST services correctly, the RESTEasy scan in JBoss must be disabled. These instructions explain how to do this.

1. Use a zip extraction utility such as 7-Zip, Winzip, or Archive to view the contents of the `<your jboss installation directory>/standalone/deployments/pentaho.war` file. Do not unzip the `pentaho.war` file, just view its contents.
2. Navigate to the `WEB-INF` directory in the `pentaho.war` file and open the `web.xml` file in a text editor.
3. At the end of the <context-param> tags, add this code.

```
<context-param>
             <param-name>resteasy.scan</param-name>
             <param-value>false</param-value>
        </context-param>
        <context-param>
             <param-name>resteasy.scan.resources</param-name>
             <param-value>false</param-value>
        </context-param>
        <context-param>
             <param-name>resteasy.scan.providers</param-name>
             <param-value>false</param-value>
</context-param>
```

4. In the `web.xml` file, change the BA Server fully qualified URL. To learn how, read the Change the BA Server Fully Qualified URL instructions.
5. Save the changes and close the file.
6. The zip extraction utility that you used might show a prompt that asks whether you would like to update the file in the `pentaho.war` archive. If this happens, confirm that you would like to do this.

Set the Location of the pentaho-solutions Directory

To deploy JBoss correctly, Pentaho recommends that you define the location of the Pentaho solutions directory in the `web.xml` file. These instructions explain how to do this.

1. If you have not done so already, use a zip extraction utility such as 7-Zip, Winzip, or Archive to view the contents of the `jboss/standalone/deployments/pentaho.war` file. Do not unzip the `pentaho.war` file, just view its contents.
2. Navigate to the `WEB-INF` directory in the `pentaho.war` file and open the `web.xml` file in a text editor.
3. Locate the following <context-param> tags.

```
<context-param>
      <param-name>solution-path</param-name>
      <param-value></param-value>
 </context-param>
```

4. Set the paramater value of the solution-path to the pentaho-solutions path. An example of the code is below.

```
<context-param>
     <param-name>solution-path</param-name>
     <param-value>/home/pentaho/server/biserver-ee/pentaho-solutions</param-
value>
 </context-param>
```

5. Save the changes and close the file.

Increase JBoss Default Memory Settings

Before you deploy the BA Server, modify the JBoss startup script to match the BA Server's memory resource requirements. If this step is not performed, the BA Server will not start.

1. Use a text editor to open the standalone configuration file. The file you open depends on your operating system.
   - **Microsoft Windows:** `<your jboss installation directory>/bin/standalone-conf.bat`
   - **Linux:** `<your jboss installation directory>/bin/standalone.conf`
4. Change the following code from `-Xms1303m -Xmx1303m -XX:MaxPermSize=256m` to this:

```
-Xms4096m -Xmx6144m -XX:MaxPermSize=256m
```

5. Save the changes and close the file.

Add JBoss Logging

You can add Pentaho application level logging to the JBoss logging subsystem. This is an optional step.

1. In the **pentaho.war**, go to the `WEB-INF` directory and open the `jboss-deployment-structure.xml` file.
2. Under the `<deployment>` tag add this code.

```
<exclusions>
    <module name="org.apache.log4j"/>
</exclusions>
```

3. Save and close the file, then exit from **pentaho.war**.
4. Open the `standalone.xml` file in the `<your jboss directory>/standalone/configuration` directory.
5. Under the `</extensions>` tag add this code.

```
 <system-properties>
     <property name="org.jboss.as.logging.per-deployment" value="false"/>
  </system-properties>
```

6. Find the `<console-handler ...>` or `<file-handler ...>` sections and add the following two handlers.

```
<console-handler name="PENTAHOCONSOLE">
<level name="ALL"/>
</console-handler>
   <file-handler name="PENTAHOFILE">
    <file relative-to="jboss.server.log.dir" path="pentaho.log"/>
    <append value="false"/>
</file-handler>
```

7. Under the file handlers section, there is a section containing `<logger>` tags. Configure the `<root-logger>` handler to use these handlers.

```
<handler name="PENTAHOCONSOLE"/>
<handler name="PENTAHOFILE"/>
```

8. Add the following loggers above `<root-logger>`.

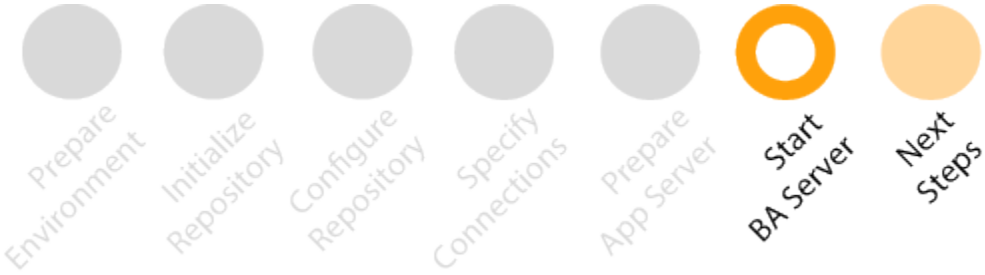
```
<logger category="org.hybernate" use-parent-handlers="false">
    <level name="ERROR"/>
        <handlers>
            <handler name="PENTAHOFILE"/>
            <handler name="PENTAHOCONSOLE"/>
        </handlers>
</logger>
<logger category="net.sf.ehcache" use-parent-handlers="false">
    <level name="ERROR"/>
        <handlers>
            <handler name="PENTAHOFILE"/>
            <handler name="PENTAHOCONSOLE"/>
        </handlers>
   </logger>
<logger category="org.quartz" use-parent-handlers="false">
    <level name="ERROR"/>
        <handlers>
            <handler name="PENTAHOFILE"/>
            <handler name="PENTAHOCONSOLE"/>
</handlers>
</logger>
    <logger category="org.springframework" use-parent-handlers="false">
    <level name="ERROR"/>
        <handlers>
            <handler name="PENTAHOFILE"/>
            <handler name="PENTAHOCONSOLE"/>
```

```
        </handlers>
</logger>
<logger category="org.springframework.security" use-parent-handlers="false">
    <level name="ERROR"/>
    <handlers>
        <handler name="PENTAHOFILE"/>
        <handler name="PENTAHOCONSOLE"/>
    </handlers>
</logger>
<logger category="org.pentaho" use-parent-handlers="false">
    <level name="ERROR"/>
    <handlers>
        <handler name="PENTAHOFILE"/>
        <handler name="PENTAHOCONSOLE"/>
    </handlers>
</logger>
<logger category="com.pentaho" use-parent-handlers="false">
    <level name="ERROR"/>
    <handlers>
        <handler name="PENTAHOFILE"/>
        <handler name="PENTAHOCONSOLE"/>
    </handlers>
</logger>
<logger category="org.jfree.JCommon" use-parent-handlers="false">
    <level name="ERROR"/>
    <handlers>
        <handler name="PENTAHOFILE"/>
        <handler name="PENTAHOCONSOLE"/>
    </handlers>
</logger>
<logger category="org.apache.jackrabbit.core.security.authentication.
AbstractLoginModule" use-parent-handlers="false">
    <level name="ERROR"/>
    <handlers>
        <handler name="PENTAHOFILE"/>
        <handler name="PENTAHOCONSOLE"/>
    </handlers>
</logger>
<logger category="RepositoryImportLog" use-parent-handlers="false">
    <level name="INFO"/>
```

```
    <handlers>
        <handler name="PENTAHOFILE"/>
        <handler name="PENTAHOCONSOLE"/>
    </handlers>
</logger>
```

9. Save and close the file.

Start BA Server



After you've complete the tasks in the Prepare Environment, Initialize Repository, Specify Connections, and Prepare Web Server steps, the BA Server's `pentaho.war` file is ready to be deployed. The way that tasks for deployment are performed vary slightly depending on whether you deploy the `pentaho.war` file on the Tomcat or JBoss web application servers.

Once the `pentaho.war` file has been deployed, manually start the BA Server application.

Modify Tomcat Startup Script

The Tomcat startup script must be modified to include the `CATALINA_OPTS` variable. `CATALINA_OPTS` indicates the amount of memory to allocate. It also indicates where Pentaho licenses are installed. Specific instructions on how to modify the startup script depend on your operating system.

Modify the Tomcat Windows Startup Script

1. Make sure the Tomcat web application server is not running by starting the Windows **Task Manager** and looking for **Tomcat** in the **Applications** tab. If the server is running, stop it.
2. Use a text editor to open the `startup.bat` file, which is in the `bin` subdirectory of the Tomcat home directory.
3. Add this line directly before the `call "%EXECUTABLE%" start %CMD_LINE_ARGS%` line, which is located near the end of the file. `set CATALINA_OPTS=-Xms4096m -Xmx6144m -XX:MaxPermSize=256m -Dsun.rmi.dgc.client.gcInterval=3600000 -Dsun.rmi.dgc.server.gcInterval=3600000 -Dpentaho.installed.licenses.file=%PENTAHO_INSTALLED_LICENSE_PATH%`
4. Save and close the file.

Modifying the Tomcat Linux Startup Script

1. Make sure the Tomcat web application server is not running by opening a **Terminal** window and typing `ps -A` at the prompt. If the server is running, stop it.

2. Use a text editor to open the `startup.sh` file, which is in the `bin` subdirectory of the Tomcat home directory.
3. Add this line directly before the `exec "$PRGDIR"/"$EXECUTABLE" start "$@"` line near the end of the file. `export CATALINA_OPTS="-Xms4096m -Xmx6144m -XX:MaxPermSize=256m -Dsun.rmi.dgc.client.gcInterval=3600000 -Dsun.rmi.dgc.server.gcInterval=3600000 -Dpentaho.installed.licenses.file=$PENTAHO_INSTALLED_LICENSE_PATH"`
4. Save and close the file.

Modify JBoss Startup Script

The JBoss startup script must be modified to include the `JAVA_OPTS` variable. `JAVA_OPTS` indicates the amount of memory to allocate. It also indicates where Pentaho licenses are installed. Specific instructions on how to modify the startup script depend on your operating system.

Modify the JBoss Windows Startup Script

1. Make sure the JBoss web application server is not running by starting the Windows **Task Manager** and looking for **JBoss** in the **Applications** tab. If the server is running, stop it.
2. Use a text editor to open the `standalone.bat` file, which is located in the JBoss `bin` directory.
3. Add this line below the `JAVA_OPTS IF` statement. It should be outside of the brackets and not part of the `IF` statement. `set JAVA_OPTS=%JAVA_OPTS% -Xms4096m -Xmx6144m -XX:MaxPermSize=256m -Dpentaho.installed.licenses.file=%PENTAHO_INSTALLED_LICENSE_PATH%`
4. Save and close the file.

Modifying the JBoss Linux Startup Script

1. Make sure the JBoss web application server is not running by opening a **Terminal** window and typing `ps -A` at the prompt. If the server is running, stop it.
2. Use a text editor to open the `standalone.conf` file. The file is located in the `bin` subdirectory of your JBoss home directory.
3. Modify the `Xms` memory settings in the `JAVA_OPTS` line to be at least 4096 MB or more, if you have the resources and are concerned with performance. Change the `Xmx` value to at least 6144 MB.
4. Add the following options to the `JAVA_OPTS` line: `-Djava.awt.headless=true -Djava.io.tmpdir=/tmp/ -Dpentaho.installed.licenses.file=$PENTAHO_INSTALLED_LICENSE_PATH`

```
# Specify options to pass to the Java VM.
    if [ "x$JAVA_OPTS" = "x" ]; then
        JAVA_OPTS="-Xms4096m \
        -Xmx6144m \
        -XX:MaxPermSize=256m \
        -Dsun.rmi.dgc.client.gcInterval=3600000 \
        -Dsun.rmi.dgc.server.gcInterval=3600000 \
```

```
        <b>-Djava.awt.headless=true \
        -Djava.io.tmpdir=/tmp/ \
        -Dpentaho.installed.licenses.file=$PENTAHO_INSTALLED_LICENSE_PATH
```

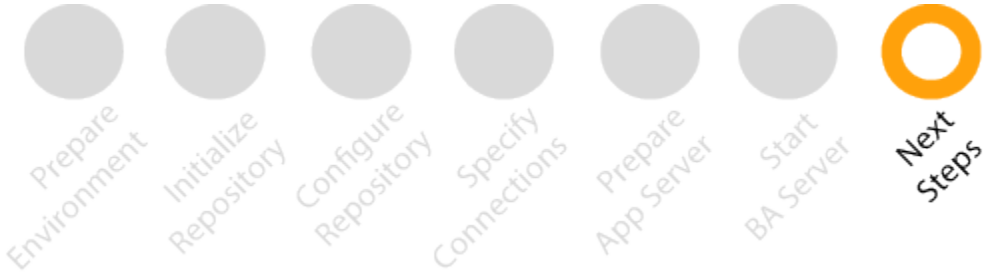
You may need to adjust these settings for your environment. For instance, if you do not have a `/tmp/` directory, you may want to change that setting to `/var/tmp/` or some other location.

5. Save and close the file.

Start BA Server

1. Run the startup script for your web application server by launching one these files.
   - **Windows Tomcat:** Launch the `startup.bat` file. The startup.bat file is in the Tomcat `bin` directory.
   - **Linux Tomcat:** Launch the `startup.bat` file. The startup.sh file is in the Tomcat `bin` directory.
   - **Windows JBoss:** Launch the `standalone.bat` file. The startup.bat file is in the JBoss `bin` directory.
   - **Linux JBoss:** Launch the `standalone.sh` file. The startup.bat file is in the JBoss `bin` directory.
6. Open a web browser and enter this URL: `http://localhost:8080/pentaho`. The **User Console Log On** window appears. Note that you will be prompted to install a license. Information on how to do that appears in the Set Up BA Server instructions.

Next Steps



Now that you've installed the BA Server, do two things.

- Install the BA design tools, so you can generate models and reports.
- Configure the BA Server and design tools so you can install licenses, set up datasources, and choose a security method, and more. You must install the license to log into the BA Server.

Note: If you have installed the BA Server so that you can migrate content from the old system to this one, make sure that your license keys have been installed, then view the Upgrade BA System instructions.

Learn More

- Web-Based Data Analysis, Reports, and Dashboards Tutorial using the User Console