pentaho

Version 5.4

Use Manual BA 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 460

5950 Hazeltine National Drive Orlando, FL 32822

Phone: +1 407 812-OPEN (6736)

Fax: +1 407 517-4575

http://www.pentaho.com

Sales Inquiries: sales@pentaho.com

Introduction 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, MS SQL Server, 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.

- Prepare Repository: Provides information about how to run DDL scripts that create tables for the BA Repository. Also provides information about how to configure the BA Repository on your selected database.

- **Prepare JBoss Connections and 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

This section explains how to prepare your environment for the installation process. Select your operating system to begin.

- Windows
- Linux and Mac

Prepare Your Windows Environment for Installation

Create Windows Directory Structure

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

```
pentaho\server\biserver-ee
```

3. Verify that you have the appropriate permissions to read, write, and execute commands in the directories you created.

Install Java

Install a supported version of Java.

1. Check the Supported Technologies list to see which version of Java Pentaho supports.
2. Download the supported version of the JRE or JDK from the Oracle site and install it.

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. It also stores metadata and models for reports that you create. You can host the BA Repository on these databases.

- PostgreSQL

- MySQL
- Oracle
- MS SQL Server

To install the BA Repository's host database, do these things.

1. Check the Supported Technologies section to determine which versions of databases Pentaho supports.
2. Download and install the database of your choice.
3. Verify that the database is installed correctly.

Download and Unpack Installation Files

The Pentaho BA Server software, data files, and examples are stored in pre-packaged `.war` and `.zip` files.

These files need to be manually placed into the correct directories.

Download Files

Download the following installation and plug-in files from the Pentaho Customer Support Portal in the manual build folder.

Component	Zip File
BA Server Installation File	`biserver-manual-ee-5.0.0-dist.zip`
Dashboard Designer Plugin	`pdd-plugin-ee-5.0.0-dist.zip`
Interactive Reporting Plugin	`pir-plugin-ee-5.0.0-dist.zip`
Mobile Plugin	`pentaho-mobile-plugin-5.0.0-dist.zip`
Pentaho Analyzer Plugin	`paz-plugin-ee-5.0.0-dist.zip`

Unpack Files

1. Unzip the BA Server Installation file.
2. To unpack the file, run `install.bat`. The **IZPak** window appears.
3. Read the license agreement, select **I accept the terms of this license agreement**, and click **Next**.
4. In the **Select the installation path** text box, enter the place where you want to create the pentaho directory, then click **Next**.
5. A message indicating that a target directory will be created appears. Click **OK**.
6. When the installation progress is complete, click **Quit**.

Put Files in Directories

1. Navigate to the pentaho directory where you unpacked the files, unzip the zip files and place them in the appropriate directories listed below.

File	Unzip the Contents of the File to This Directory
license-installer.zip	`pentaho\`
pentaho-data.zip	`pentaho\server\biserver-ee`
pentaho-solutions.zip	`pentaho\server\biserver-ee`

2. Copy these files to the following directories.

File	Copy Files to This Directory
pentaho.war	• **Tomcat:** `pentaho\server\biserver-ee\<your tomcat installation directory>\webapps` • **JBoss:** `pentaho\server\biserver-ee\<your jboss installation directory>\standalone\deployments`
pentaho-style.war	• **Tomcat:** `pentaho\server\biserver-ee\<your tomcat installation directory>\webapps` • **JBoss:** `pentaho\server\biserver-ee\<your jboss installation directory>\standalone\deployments`
PentahoBIPlatform_OSS_Licenses.html	`pentaho\server\biserver-ee`

Unpack and Unzip Plugin Files

Do the following for each of the plugin files.

1. To unpack a file, run the `install.bat` script. The **IZPak** window appears.
2. Read the license agreement, select **I accept the terms of this license agreement**, and click **Next**.
3. In the **Select the installation path** text box, enter the pentaho\server\biserver-ee\pentaho-solutions\ system directory, then click **Next**.
4. A message appears. Click **YES**.
5. When the installation progress is complete, click **Quit**.

Verify Directory Structure

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-style.war
- pentaho\server\biserver-ee\data
- pentaho\server\biserver-ee\pentaho-solutions
- pentaho\server\biserver-ee\pentaho-solutions\system\analyzer
- pentaho\server\biserver-ee\pentaho-solutions\system\dashboards
- pentaho\server\biserver-ee\pentaho-solutions\system\pentaho-interactive-reporting
- pentaho\server\biserver-ee\pentaho-solutions\system\pentaho-mobile-plugin
- pentaho\server\biserver-ee\pentaho-solutions\systems\default-content\samples.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\analyzer
- pentaho\server\biserver-ee\pentaho-solutions\system\dashboards
- pentaho\server\biserver-ee\pentaho-solutions\system\pentaho-interactive-reporting
- pentaho\server\biserver-ee\pentaho-solutions\system\pentaho-mobile-plugin
- pentaho\server\biserver-ee\pentaho-solutions\system\default-content\samples.zip

Set Environment Variables

Set the `PENTAHO_JAVA_HOME` and `PENTAHO_INSTALLED_LICENSE_PATH` environment variables. If you do not set these variables, Pentaho will not start correctly.

1. Set the path of the `PENTAHO_JAVA_HOME` variable to the path of your Java installation, like this.

```
PENTAHO_JAVA_HOME=C:\Program Files\Java\jdk7
```

2. Set the path of the `PENTAHO_INSTALLED_LICENSE_PATH` variable to the path that contains your installed licenses, like this.

```
PENTAHO_INSTALLED_LICENSE_PATH=C:\Users\pentaho\.pentaho\.installedLicenses.xml
```

3. Verify the variables have been properly set.

Next Step

You've finished preparing your environment. Go to Configure Your Repository Database to continue.

Prepare Your Linux and Mac Environment for Installation

Create the Pentaho User

Create a pentaho user account that has administrative privileges. You will use this account to complete the rest of the installation instructions.

1. Create an administrative user on computer that will host the BA Server and name it **pentaho**.
2. Verify that you have the appropriate permissions to read, write, and execute commands in the pentaho user's home directory.

Install Java

Install a supported version of Java.

1. Check the Supported Technologies list to see which version of Java Pentaho supports.
2. Download the supported version of the JRE or JDK from the Oracle site and install it.

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.

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.

Install the BA Repository Host Database

The BA Repository houses data needed for Pentaho tools to provide scheduling and security functions. The repository also stores metadata and models for reports that you create. You can choose to host the BA Repository on these databases.

- PostgreSQL
- MySQL
- Oracle
- MS SQL Server

To install the BA Repository's host database, do these things.

1. Check the Supported Technologies section to determine which versions of the databases Pentaho supports.
2. Download and install the database of your choice.
3. Verify that the BA Repository database is installed correctly.

Download and Unpack Installation Files

The Pentaho BA Server software, data files, and examples are stored in pre-packaged `.war` and `.zip` files. These files need to be manually placed into the correct directories.

Download Files

Download the following installation and plug-in files from the Pentaho Customer Support Portal in the manual build folder.

Component	Zip File
BA Server Installation File	`biserver-manual-ee-5.0.0-dist.zip`
Dashboard Designer Plugin	`pdd-plugin-ee-5.0.0-dist.zip`
Interactive Reporting Plugin	`pir-plugin-ee-5.0.0-dist.zip`
Mobile Plugin	`pentaho-mobile-plugin-5.0.0-dist.zip`
Pentaho Analyzer Plugin	`paz-plugin-ee-5.0.0-dist.zip`

Unpack Files

1. Unzip the BA Server Installation file.
2. To unpack the file, run `install.sh`. The **IZPak** window appears.

NOTE:

If you are unpacking the file in a non-graphical environment, open a **Terminal** or **Command Prompt** and type `java -jar installer.jar -console` and follow the instructions presented in the window.

1. Read the license agreement, select **I accept the terms of this license agreement**, and click **Next**.
2. In the **Select the installation path** text box, enter the place where you want to create the pentaho directory, then click **Next**.
3. A message indicating that a target directory will be created appears. Click **OK**.
4. When the installation progress is complete, click **Quit**.

Put Files in Directories

1. Navigate to the pentaho directory where you unpacked the files, unzip the zip files and place them in the appropriate directories listed below.

File	Unzip the Contents of the File to This Directory
license-installer.zip	`pentaho/`
pentaho-data.zip	`pentaho/server/biserver-ee`
pentaho-solutions.zip	`pentaho/server/biserver-ee`

2. Copy these files to the following directories.

File	Copy Files to This Directory
pentaho.war	• **Tomcat:** `pentaho/server/biserver-ee/<your tomcat installation directory>/webapps` • **JBoss:** `pentaho/server/biserver-ee/<your jboss installation directory>/standalone/deployments`
pentaho-style.war	• **Tomcat:** `pentaho/server/biserver-ee/<your tomcat installation directory>/webapps` • **JBoss:** `pentaho/server/biserver-ee/<your jboss installation directory>/standalone/deployments`
PentahoBIPlatform_OSS_Licenses.html	`pentaho/server/biserver-ee`

Unpack and Unzip Plugin Files

Do the following for each of the plugin files.

1. To unpack the file, run `install.sh`. The **IZPak** window appears.

NOTE:

If you are unpacking the file in a non-graphical environment, open a **Terminal** or **Command Prompt** and type `java -jar installer.jar -console` and follow the instructions presented in the window.

1. Read the license agreement, select **I accept the terms of this license agreement**, and click **Next**.
2. In the **Select the installation path** text box, enter the pentaho\server\biserver-ee\pentaho-solutions\ system directory, then click **Next**.
3. A message appears. Click **YES**.
4. When the installation progress is complete, click **Quit**.

Verify Directory Structure

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-style.war
- pentaho/server/biserver-ee/data
- pentaho/server/biserver-ee/pentaho-solutions
- pentaho/server/biserver-ee/pentaho-solutions/system/analyzer
- pentaho/server/biserver-ee/pentaho-solutions/system/dashboards
- pentaho/server/biserver-ee/pentaho-solutions/system/pentaho-interactive-reporting
- pentaho/server/biserver-ee/pentaho-solutions/system/pentaho-mobile-plugin
- pentaho/server/biserver-ee/pentaho-solutions/systems/default-content/samples.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/analyzer
- pentaho/server/biserver-ee/pentaho-solutions/system/dashboards
- pentaho/server/biserver-ee/pentaho-solutions/system/pentaho-interactive-reporting
- pentaho/server/biserver-ee/pentaho-solutions/system/pentaho-mobile-plugin
- pentaho/server/biserver-ee/pentaho-solutions/system/default-content/samples.zip

Set Environment Variables

Set the `PENTAHO_JAVA_HOME` and `PENTAHO_INSTALLED_LICENSE_PATH` environment variables. If you do not set these variables, Pentaho will not start correctly.

NOTE:

If you are using a JRE, set the `JRE_HOME` home environment variable as well.

1. Set the path of the `PENTAHO_JAVA_HOME` variable to the path of your Java installation, like this.

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

2. Set the path of the `PENTAHO_INSTALLED_LICENSE_PATH` variable to the path of the installed licenses, like this.

```
export PENTAHO_INSTALLED_LICENSE_PATH=/home/pentaho/.pentaho/.
installedLicenses.xml
```

3. Log out and in again, then verify the variables have been properly set.

Advanced Linux and Mac 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 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.

Next Step

You've finished preparing your environment. Go to Configure Your Repository Database to continue.

Configure Your Repository Database

Select the database that you are using as the solution repository.

- PostgreSQL
- MySQL
- Oracle
- MS SQL Server

Use PostgreSQL as Your Repository Database

Before you prepare your Business Analytics (BA) Repository, complete the tasks in Prepare Environment.

The BA Repository resides on the database that you installed during the Prepare Environment step, and consists of four repositories: *Jackrabbit*, *Quartz*, *Hibernate,* and *Pentaho Operations Mart*.

- *Jackrabbit* contains the solution repository, 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.
- *Pentaho Operations Mart* reports on system usage and performance.

Initialize PostgreSQL BA Repository Database

To initialize PostgreSQL so that it serves as the BA Repository, you will need to run a few SQL scripts to create the Hibernate, Quartz, Jackrabbit (JCR), and Pentaho Operations Mart 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.

These sections take you through the steps to initialize the PostgreSQL BA repository database.

Change Default Passwords

We recommend that you change the default passwords in the SQL script files. If you are evaluating Pentaho, you might want to skip this step.

If you do decide to make the databases more secure, use any 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`
- `pentaho/server/biserver-ee/data/postgresql/pentaho_mart_postgresql.sql`

Run SQL Scripts

Once you change the passwords, you will need to run these SQL scripts. You will need administrator permissions on the server in order to run these scripts. The process for running SQL scripts is the same for Windows or Linux machines. The list of SQL scripts is shown in the table below.

CAUTION:

If you have a different port or different password, make sure that you change the password and port number in these examples to match the ones in your configuration.

Run these scripts from the **PSQL Console** window in the **pgAdminIII** tool.

SQL Scripts

Action	SQL Script
Create Quartz	`\i <your filepath>/data/postgresql/create_quartz_postgresql.sql`
Create Hibernate repository	`\i <your filepath>/data/postgresql/create_repository_postgresql.sql`
Create Jackrabbit	`\i <your filepath>/data/postgresql/create_jcr_postgresql.sql`
Create Pentaho Operations mart	`\i <your filepath>/data/postgresql/pentaho_mart_postgresql.sql`

Verify PostgreSQL Initialization

After you run the scripts, this list will help you verify that databases and user roles have been created.

1. Open the **pgAdminIII** tool.
2. Verify that you can log in as **hibuser**.
3. Once logged in, make sure that the Quartz, Jackrabbit (JCR), Hibernate, and Pentaho Operations mart databases are present.
4. Exit from the **pgAdminIII**.

Configure PostgreSQL BA Repository Database

Now that you have initialized your repository database, you will need to configure Quartz, Hibernate, Jackrabbit, and Pentaho Operations Mart for a PostgreSQL database.

By default, the examples in this section are for a PostgreSQL database that runs on port 5432. The default password is also in these examples. If you have a different port or different password, complete all of the instructions in these steps.

CAUTION:

If you have a different port or different password, make sure that you change the password and port number in these examples to match the ones in your configuration.

Set Up Quartz on PostgreSQL BA Repository Database

Event information, such as scheduled reports, is stored in the Quartz JobStore. During the installation process, you must indicate where the **JobStore** is located. You do this by modifying the `quartz.properties` file.

1. Open the `pentaho/server/biserver-ee/pentaho-solutions/system/quartz/quartz.properties` file in any text editor.
2. Locate the `#_replace_jobstore_properties` section and set the `org.quartz.jobStore.driverDelegateClass` as shown here.

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

3. Locate the `# Configure Datasources` section and 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.

Set Hibernate Settings for PostgreSQL

Modify the hibernate settings file to specify where Pentaho should find the BA Repository's hibernate config file. The hibernate config file specifies driver and connection information, as well as dialects and how to handle connection closes and timeouts. The files in this section are located in the `pentaho/server/biserver-ee/pentaho-solutions/system/hibernate` directory.

1. Open the `hibernate-settings.xml` file in a text editor. Find the `<config-file>` tags and confirm that it is configured for PostgreSQL.

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

2. Save the file if you made changes, and close the file.
3. Open the `postgresql.hibernate.cfg.xml` file in a text editor.
4. Make sure that the password and port number match the ones you specified in your configuration. Make changes if necessary, then save and close the file.

Modify Jackrabbit Repository Information for PostgreSQL

There are parts of code that you will need to alter in order to change the default jackrabbit repository to PostgreSQL.

1. Navigate to the `pentaho/server/biserver-ee/pentaho-solutions/system/jackrabbit` and open the `repository.xml` file with any text editor.
2. Following the table below, locate and verify or change the code so that the PostgreSQL lines are **not** commented out, but the MySQL, Oracle, and MS SQL Server lines **are** commented out.

CAUTION:

If you have a different port or different password, make sure that you change the password and port number in these examples to match the ones in your configuration.

Item:	Code Section:

Repository	`<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"/>`
	`  ...`
	`  </FileSystem>`

DataStore	`<DataStore class="org.apache.jackrabbit.core.data.db.DbDataStore">`
	`  <param name="url" value="jdbc:postgresql://localhost:5432/jackrabbit"/>`
	`  ...`
	`  </DataStore>`

Workspaces	`<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"/>`
	`  ...`
	`  </FileSystem>`

PersistenceManager
(1st part)
	`<PersistenceManager class="org.apache.jackrabbit.core.persistence.bundle.PostgreSQLPersistenceManager">`
	`  <param name="url" value="jdbc:postgresql://localhost:5432/jackrabbit"/>`
	`  ...`
	`  </PersistenceManager>`

Versioning	`<FileSystem class="org.apache.jackrabbit.core.fs.db.DbFileSystem">`

Item:	Code Section:

	`<param name="driver" value="org.postgresql.Driver"/>`
	`<param name="url" value="jdbc:postgresql://localhost:5432/jackrabbit"/>`
	`...`
	`</FileSystem>`

PersistenceManager
(2nd part)
	`<PersistenceManager class="org.apache.jackrabbit.core.persistence.bundle.PostgreSQLPersistenceManager">`
	`<param name="url" value="jdbc:postgresql://localhost:5432/jackrabbit"/>`
	`...`
	`</PersistenceManager>`

Configure Pentaho Operations Mart

If you changed the user or password in the previous sections, make sure that you also update that information for the Pentaho Operations Mart repository.

Are you using JBoss?

If you are using JBoss, you can skip the Tomcat section and move on to Prepare JBoss Connections and Web App Servers.

Perform Tomcat-Specific Connection Tasks

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.

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.

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

Download Drivers and Install with the JDBC Distribution Tool

To connect to a database, including the BA Repository or DI Repository database, you will need to download and install a JDBC driver to the appropriate places for Pentaho components as well as on the the web

application server that contains the Pentaho Server(s). Fortunately, the JDBC Distribution Tool makes this process easy.

NOTE:

Due to licensing restrictions, Pentaho cannot redistribute some third-party database drivers. This is why you have to download the file yourself and install it yourself.

1. Download a JDBC driver JAR from your database vendor or a third-party driver developer.
2. Copy the JDBC driver JAR you just downloaded to the `pentaho/jdbc-distribution` directory.
3. Open a cmd prompt or shell tool, navigate to the `pentaho/jdbc-distribution` directory and enter one of the following:

Windows:

```
distribute-files.bat <name of JDBC driver JAR>
```

Linux:

```
./ distribute-drivers.sh
```

1. If you have run this utility as part of the installation process, you are done. Go to the next step of the installation instructions.
2. If you have run this utility so that you can connect to a new repository, restart the BA or DI Server and Design tools, then try to connect to the new repository. If you cannot connect, verify that the drivers are installed as shown in this table. Restart your Pentaho Server(s) and Client tools.

List of Products and Corresponding Locations for JDBC Drivers

Server or Design Tool	Directory
Business Analytics (BA) Server	pentaho/server/biserver-ee/tomcat/lib
Data Integration (DI) Server	pentaho/server/data-integration-server/tomcat/lib
Pentaho Data Integration (Spoon)	pentaho/design-tools/data-integration/lib
Pentaho Report Designer (PRD)	pentaho/design-tools/report-designer/lib/jdbc
Pentaho Aggregation Designer (PAD)	pentaho/design-tools/aggregation-designer/drivers
Pentaho Schema Workbench (PSW)	pentaho/design-tools/schema-workbench/drivers
Pentaho Metadata Editor (PME)	pentaho/design-tools/metadata-editor/libext/JDBC

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.

CAUTION:

If you have a different port, password, user, driver class information, or IP address, make sure that you change the password and port number in these examples to match the ones in your configuration environment.

1. Consult your database documentation to determine the JDBC class name and connection string for your BA Repository database.
2. Go to the `biserver-ee/tomcat/webapps/pentaho/META-INF` directory and open the `context.xml` file with any file editor.
3. Comment out the resource references that refer to databases other than PostgreSQL such as MySQL, MS SQL Server, and Oracle. Then, add the following code to the file if it does not already exist. Be sure to 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/PDI_Operations_Mart"/>
```

4. Make sure that the `validationQuery` variable for your database is set to this: `validationQuery="select 1"`
5. Save the `context.xml` file, then close it.
6. To make sure that the changes that you made in the context.xml file take effect when Tomcat is started, navigate to the `tomcat/conf/Catalina` 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.

Next Steps

Now it is time to start your server.

Use MySQL as Your Repository Database

Before you prepare your Business Analytics (BA) Repository, complete the tasks in Prepare Environment.

The BA Repository resides on the database that you installed during the Prepare Environment step, and consists of four repositories: *Jackrabbit*, *Quartz*, *Hibernate,* and *Pentaho Operations Mart*.

- *Jackrabbit* contains the solution repository, 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.
- *Pentaho Operations Mart* reports on system usage and performance.

Initialize MySQL BA Repository Database

To initialize MySQL so that it serves as the BA Repository, you will need to run a few SQL scripts to create the Hibernate, Quartz, Jackrabbit (JCR), and Pentaho Operations Mart 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.

The next few sections take you through the steps to initialize the MySQL BA repository database.

Change Default Passwords

We recommend that you change the default passwords in the SQL script files. If you are evaluating Pentaho, you might want to skip this step.

If you do decide to make the databases more secure, use any 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`
- `pentaho/server/biserver-ee/data/mysql5/pentaho_mart_mysql.sql`

Run SQL Scripts

Once you change the passwords, you will need to run these SQL scripts. You will need administrator permissions on the server in order to run these scripts. The process for running SQL scripts is the same for Windows or Linux machines. The list of SQL scripts is shown in the table below.

CAUTION:

If you have a different port or different password, make sure that you change the password and port number in these examples to match the ones in your configuration.

Run these scripts from the **MySQL Command Prompt** window or from **MySQL Workbench**.

Action	SQL Script
Create Quartz	`> source <your filepath>\create_quartz_mysql.sql`
Create Hibernate repository	`> source <your filepath>\create_repository_mysql.sql`
Create Jackrabbit	`> source <your filepath>\create_jcr_mysql.sql`
Create Pentaho Operations mart	`> source <your filepath>\pentaho_mart_mysql.sql`

Verify MySQL Initialization

After you run the scripts, this list will help you verify that databases and user roles have been created.

1. Open the **MySQL Workbench** tool. **MySQL Workbench** is freely available at the MySQL development site.
2. Log in as **hibuser**.
3. Make sure that the Quartz, Jackrabbit (JCR), Hibernate, and Pentaho Operations Mart databases are present.
4. Exit from the **MySQL Workbench**.

Configure MySQL BA Repository Database

Now that you have initialized your repository database, you will need to configure Quartz, Hibernate, Jackrabbit, and Pentaho Operations Mart 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.

CAUTION:

If you have a different port or different password, make sure that you change the password and port number in these examples to match the ones in your configuration.

Set Up Quartz on MySQL BA Repository Database

Event information, such as scheduled reports, is stored in the Quartz `JobStore`. During the installation process, you must indicate where the `JobStore` is located, by modifying the `quartz.properties` file.

1. Open the `pentaho/server/biserver-ee/pentaho-solutions/system/quartz/quartz.properties` file in any text editor.

2. Locate the `#_replace_jobstore_properties` section and set the `org.quartz.jobStore.driverDelegateClass` as shown here.

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

3. Locate the `# Configure Datasources` section and 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.

Set Hibernate Settings for MySQL

Modify the hibernate settings file to specify where Pentaho should find the BA Repository's hibernate config file.

NOTE:

The hibernate config file specifies driver and connection information, as well as dialects and how to handle connection closes and timeouts.

The files in this section are located in the `pentaho/server/biserver-ee/pentaho-solutions/system/hibernate` directory.

1. Open the `hibernate-settings.xml` file in a text editor. Find the `<config-file>` tags and change `postgresql.hibernate.cfg.xml` to `mysql5.hibernate.cfg.xml` as shown.
From:

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

To:

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

2. Save and close the file.
3. Open the `mysql5.hibernate.cfg.xml` file in a text editor.
4. Make sure that the password and port number match the ones you specified in your configuration. Make changes if necessary, then save and close the file.

Replace Default Version of Audit Log File with MySQL Version

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.

1. Locate the `pentaho-solutions/system/dialects/mysql5/audit_sql.xml` file.
2. Copy it into the `pentaho-solutions/system` directory.

Modify Jackrabbit Repository Information for MySQL

There are parts of code that you will need to alter in order to change the default JCR repository to MySQL.

1. Navigate to the `pentaho/server/biserver-ee/pentaho-solutions/system/jackrabbit` and open the `repository.xml` file with any text editor.
2. Following the table below, locate and change the code so that the MySQL lines are **not** commented out, but the PostgreSQL, MS SQL Server, and Oracle lines **are** commented out.

CAUTION:

If you have a different port or different password, make sure that you change the password and port number in these examples to match the ones in your configuration.

Item:	Code Section:

Repository	`<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"/>`
	…
	`</FileSystem>`

DataStore	`<DataStore class="org.apache.jackrabbit.core.data.db.DbDataStore">`
	`<param name="url" value="jdbc:mysql://localhost:3306/jackrabbit"/>`
	…
	`</DataStore>`

Workspaces	`<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"/>`
	…
	`</FileSystem>`

PersistenceManager (1st part)	`<PersistenceManager class="org.apache.jackrabbit.core.persistence.bundle.MySqlPersistenceManager">` `<param name="url" value="jdbc:mysql://localhost:3306/`

Item:	Code Section:
	jackrabbit"/> … </PersistenceManager>
Versioning	<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"/> … </FileSystem>
PersistenceManager (2nd part)	<PersistenceManager class="org.apache.jackrabbit.core. persistence.bundle.MySqlPersistenceManager"> <param name="url" value="jdbc:mysql://localhost:3306/ jackrabbit"/> … </PersistenceManager>

Are you using JBoss?

If you are using JBoss, you can skip the Tomcat section and move on to Prepare JBoss Connections and Web App Servers.

Perform Tomcat-Specific Connection Tasks

After your repository has been configured, you must configure the web application servers to connect to the BA Repository. In this section, JDBC and JNDI connections are made to the Hibernate, Jackrabbit, and Quartz databases.

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.

The next couple of sections guide you through the process of working with the JDBC drivers and connection information for Tomcat.

Download Drivers and Install with the JDBC Distribution Tool

To connect to a database, including the BA Repository or DI Repository database, you will need to download and install a JDBC driver to the appropriate places for Pentaho components as well as on the the web application server that contains the Pentaho Server(s). Fortunately, the JDBC Distribution Tool makes this process easy.

NOTE:

Due to licensing restrictions, Pentaho cannot redistribute some third-party database drivers. This is why you have to download the file yourself and install it yourself.

1. Download a JDBC driver JAR from your database vendor or a third-party driver developer.
2. Copy the JDBC driver JAR you just downloaded to the `pentaho/jdbc-distribution` directory.
3. Open a cmd prompt or shell tool, navigate to the `pentaho/jdbc-distribution` directory and enter one of the following:

Windows:

```
distribute-files.bat <name of JDBC driver JAR>
```

Linux:

```
./ distribute-drivers.sh
```

1. If you have run this utility as part of the installation process, you are done. Go to the next step of the installation instructions.
2. If you have run this utility so that you can connect to a new repository, restart the BA or DI Server and Design tools, then try to connect to the new repository. If you cannot connect, verify that the drivers are installed as shown in this table. Restart your Pentaho Server(s) and Client tools.

List of Products and Corresponding Locations for JDBC Drivers

Server or Design Tool	Directory
Business Analytics (BA) Server	pentaho/server/biserver-ee/tomcat/lib
Data Integration (DI) Server	pentaho/server/data-integration-server/tomcat/lib
Pentaho Data Integration (Spoon)	pentaho/design-tools/data-integration/lib
Pentaho Report Designer (PRD)	pentaho/design-tools/report-designer/lib/jdbc
Pentaho Aggregation Designer (PAD)	pentaho/design-tools/aggregation-designer/drivers
Pentaho Schema Workbench (PSW)	pentaho/design-tools/schema-workbench/drivers
Pentaho Metadata Editor (PME)	pentaho/design-tools/metadata-editor/libext/JDBC

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.

CAUTION:

If you have a different port, password, user, driver class information, or IP address, make sure that you change the password and port number in these examples to match the ones in your configuration environment.

1. Consult your database documentation to determine the JDBC class name and connection string for your BA Repository database.
2. Go to the `biserver-ee/tomcat/webapps/pentaho/META-INF` directory and open the `context.xml` file with any file editor.
3. Comment out the resource references that refer to databases other than MySQL, such as PostgreSQL, MS SQL Server, and Oracle. Then, add the following code to the file if it does not already exist. Be sure to 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/PDI_Operations_Mart"/>
```

4. Make sure that the `validationQuery` variable for your database is set to this: `validationQuery="select 1"`
5. Save the `context.xml` file, then close it.

6. To make sure that the changes that you made in the context.xml file take effect when Tomcat is started, navigate to the `tomcat/conf/Catalina` 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.

Next Steps

Now it is time to start your server.

Use Oracle as Your Repository Database

Before you prepare your Business Analytics (BA) Repository, complete the tasks in Prepare Environment.

The BA Repository resides on the database that you installed during the Prepare Environment step, and consists of four repositories: *Jackrabbit*, *Quartz*, *Hibernate,* and *Pentaho Operations Mart*.

- *Jackrabbit* contains the solution repository, 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.
- *Pentaho Operations Mart* reports on system usage and performance.

Initialize Oracle BA Repository Database

To initialize Oracle so that it serves as the BA Repository, you will need to run a few SQL scripts to create the Hibernate, Quartz, Pentaho Operations mart, 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.

These sections take you through the steps to initialize the Oracle BA repository database.

Change Default Passwords

Pentaho recommends that you change the default passwords in the SQL script files. If you are evaluating Pentaho, you might want to skip this step.

If you do decide to change the passwords, use any text editor to change the passwords in these files:

- `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`
- `pentaho/server/biserver-ee/data/oracle10g/pentaho_mart_oracle.sql`

Run SQL Scripts

Once you change the passwords, you will need to run these SQL scripts. You will need administrator permissions on the server in order to run these scripts. The process for running SQL scripts is the same for Windows or Linux machines. The list of SQL scripts is shown in the table below.

CAUTION:

If you have a different port or different password, make sure that you change the password and port number in these examples to match the ones in your configuration.

Run these scripts from the **Command Prompt** window or from a **Terminal** window that runs **SQL*Plus**.

Action	SQL Script
Create Quartz	`> start <your filepath>\create_quartz_ora.sql`
Create Hibernate repository	`> start <your filepath>\create_repository_ora.sql`
Create Jackrabbit	`> start <your filepath>\create_jcr_ora.sql`
Create Pentaho Operations mart	`> start <your filepath>\pentaho_mart_oracle.sql`

Verify Oracle Initialization

After you run the scripts, this list will help you verify that databases and user roles have been created.

1. Open the Terminal or Command Prompt window that is running SQL*Plus.
2. Make sure that users have been created by running SELECT USERNAME FROM DBA_USERS.
3. If your databases do not appear, go to the beginning of these instructions and try running the scripts again.
4. Exit from the **SQL*Plus**.

Configure Oracle BA Repository Database

Now that you have initialized your repository database, you will need to configure Quartz, Hibernate, Jackrabbit, and Pentaho Operations Mart for an Oracle database.

By default, the examples in this section are for an Oracle database that runs on port 1521. The default password is also in these examples.

CAUTION:

If you have a different port or different password, make sure that you change the password and port number in these examples to match the ones in your configuration.

Set Up Quartz on Oracle BA Repository Database

Event information, such as scheduled reports, is stored in the Quartz JobStore. During the installation process, you must indicate where the **JobStore** is located, by modifying the `quartz.properties` file.

1. Open the `pentaho/server/biserver-ee/pentaho-solutions/system/quartz/quartz.properties` file in any text editor.
2. Locate the `#_replace_jobstore_properties` section and set the `org.quartz.jobStore.driverDelegateClass` as shown here.

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

3. Locate the `# Configure Datasources` section and 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.

Set Hibernate Settings for Oracle

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

The files in this section are located in the `pentaho/server/biserver-ee/pentaho-solutions/system/hibernate` directory.

1. Open the `hibernate-settings.xml` file in a text editor. Find the `<config-file>` tags and change `postgresql.hibernate.cfg.xml` to `oracle10g.hibernate.cfg.xml` as shown.
From:

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

To:

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

2. Save and close the file.
3. Open the `oracle10g.hibernate.cfg.xml` file in a text editor.
4. Make sure that the password and port number match the ones you specified in your configuration. Make changes if necessary, then save and close the file.

Replace Default Version of Audit Log File with Oracle Version

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.

1. Locate the `pentaho-solutions/system/dialects/oracle10g/audit_sql.xml` file.
2. Copy it into the `pentaho-solutions/system` directory.

Modify Jackrabbit Repository Information for Oracle

There are parts of code that you will need to alter in order to change the default jackrabbit repository to Oracle.

1. Navigate to the `pentaho/server/biserver-ee/pentaho-solutions/system/jackrabbit` and open the `repository.xml` file with any text editor.

2. Following the table below, locate and change the code so that the Oracle lines are **not** commented out, but the PostgreSQL, MS SQL Server, and MySQL lines **are** commented out.

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 in each section accordingly.

Item:	Code Section:

Repository	`<FileSystem class="org.apache.jackrabbit.core.fs.db.OracleFileSystem">`
	`  <param name="url" value="jdbc:oracle:thin:@localhost:1521/XE"/>`
	`  ...`
	`  <param name="tablespace" value="pentaho_tablespace"/>`
	`</FileSystem>`

DataStore	`<DataStore class="org.apache.jackrabbit.core.data.db.DbDataStore">`
	`  <param name="url" value="jdbc:oracle:thin:@localhost:1521/XE"/>`
	`  <param name="driver" value="oracle.jdbc.OracleDriver"/>`
	`  <param name="user" value="jcr_user"/>`
	`  <param name="password" value="password"/>`
	`  <param name="databaseType" value="oracle"/>`
	`  ...`
	`  </DataStore>`

Workspaces	`<FileSystem class="org.apache.jackrabbit.core.fs.db.OracleFileSystem">`
	`  <param name="url" value="jdbc:oracle:thin:@localhost:1521/XE"/>`
	`  ...`
	`  <param name="tablespace" value="pentaho_tablespace"/>`
	`  </FileSystem>`

Item:	Code Section:
PersistenceManager (1st part)	<PersistenceManager class="org.apache.jackrabbit.core. persistence.bundle.OraclePersistenceManager"> <param name="url" value="jdbc:oracle:thin:@localhost:1521/XE"/> <param name="driver" value="oracle.jdbc. 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="pentaho_ tablespace"/> </PersistenceManager>
Versioning	<FileSystem class="org.apache.jackrabbit.core.fs.db. OracleFileSystem"> <param name="url" value="jdbc:oracle:thin:@localhost:1521/XE"/> ... <param name="tablespace" value="pentaho_ tablespace"/> </FileSystem>
PersistenceManager (2nd part)	<PersistenceManager class="org.apache.jackrabbit.core. persistence.bundle.OraclePersistenceManager"> <param name="url" value="jdbc:oracle:thin:@localhost:1521/XE"/> <param name="driver" value="oracle.jdbc. OracleDriver"/> ... <param name="tablespace" value="pentaho_ tablespace"/> </PersistenceManager>

Are you using JBoss?

If you are using JBoss, you can skip the Tomcat section and move on to Prepare JBoss Connections and Web App Servers.

Perform Tomcat-Specific Connection Tasks

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.

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.

The next couple of sections guide you through the process of working with the JDBC drivers and connection information for Tomcat.

Download Drivers and Install with the JDBC Distribution Tool

To connect to a database, including the BA Repository or DI Repository database, you will need to download and install a JDBC driver to the appropriate places for Pentaho components as well as on the the web application server that contains the Pentaho Server(s). Fortunately, the JDBC Distribution Tool makes this process easy.

NOTE:

Due to licensing restrictions, Pentaho cannot redistribute some third-party database drivers. This is why you have to download the file yourself and install it yourself.

1. Download a JDBC driver JAR from your database vendor or a third-party driver developer.
2. Copy the JDBC driver JAR you just downloaded to the `pentaho/jdbc-distribution` directory.
3. Open a cmd prompt or shell tool, navigate to the `pentaho/jdbc-distribution` directory and enter one of the following:

Windows:

```
distribute-files.bat <name of JDBC driver JAR>
```

Linux:

```
./ distribute-drivers.sh
```

1. If you have run this utility as part of the installation process, you are done. Go to the next step of the installation instructions.
2. If you have run this utility so that you can connect to a new repository, restart the BA or DI Server and Design tools, then try to connect to the new repository. If you cannot connect, verify that the drivers are installed as shown in this table. Restart your Pentaho Server(s) and Client tools.

List of Products and Corresponding Locations for JDBC Drivers

Server or Design Tool	Directory
Business Analytics (BA) Server	pentaho/server/biserver-ee/tomcat/lib
Data Integration (DI) Server	pentaho/server/data-integration-server/tomcat/lib
Pentaho Data Integration (Spoon)	pentaho/design-tools/data-integration/lib
Pentaho Report Designer (PRD)	pentaho/design-tools/report-designer/lib/jdbc
Pentaho Aggregation Designer (PAD)	pentaho/design-tools/aggregation-designer/drivers
Pentaho Schema Workbench (PSW)	pentaho/design-tools/schema-workbench/drivers
Pentaho Metadata Editor (PME)	pentaho/design-tools/metadata-editor/libext/JDBC

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.

CAUTION:

If you have a different port, password, user, driver class information, or IP address, make sure that you change the password and port number in these examples to match the ones in your configuration environment.

1. Consult your database documentation to determine the JDBC class name and connection string for your BA Repository database.
2. Go to the `biserver-ee/tomcat/webapps/pentaho/META-INF` directory and open the `context.xml` file with any file editor.
3. Comment out the resource references that refer to databases other than Oracle, such as PostgreSQL, MS SQL Server, and MySQL. Then, add the following code to the file if it does not already exist.

   Be sure to 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="pentaho_operations_mart" username="pentaho_operations_
mart" maxWait="10000" maxIdle="5" maxActive="20" factory="org.apache.commons.
dbcp.BasicDataSourceFactory" type="javax.sql.DataSource" auth="Container"
name="jdbc/PDI_Operations_Mart"/>
```

4. Make sure that the `validationQuery` variable for your database is set to this: `validationQuery="select 1 from dual"`.
5. Save the `context.xml` file, then close it.
6. To make sure that the changes that you made in the context.xml file take effect when Tomcat is started, navigate to the `tomcat/conf/Catalina` 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 Tomcat-Specific Connection Tasks

After your repository has been configured, you must configure the web application servers to connect to the DI Repository. In this step, JDBC and JNDI connections are made to the Hibernate, Jackrabbit, and Quartz databases.

By default, the DI 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.

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

Download and Install Repository Database JDBC Drivers

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

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

NOTE:

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.

1. Copy the JDBC driver JAR that you just downloaded to the `<your tomcat installation directory>/lib` directory for the server.
2. If you are also installing Spoon, copy the JDBC driver JAR into the directory shown in the table.

List of Products and Corresponding Locations for JDBC Drivers

Server or Design Tool	Directory
Business Analytics (BA) Server	pentaho/server/biserver-ee/tomcat/lib
Data Integration (DI) Server	pentaho/server/data-integration-server/tomcat/lib
Pentaho Data Integration (Spoon)	pentaho/design-tools/data-integration/lib
Pentaho Report Designer (PRD)	pentaho/design-tools/report-designer/lib/jdbc
Pentaho Aggregation Designer (PAD)	pentaho/design-tools/aggregation-designer/drivers
Pentaho Schema Workbench (PSW)	pentaho/design-tools/schema-workbench/drivers
Pentaho Metadata Editor (PME)	pentaho/design-tools/metadata-editor/libext/JDBC

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 chosen to use an BA Repository database other than PostgreSQL.

CAUTION:

If you have a different port, password, user, driver class information, or IP address, make sure that you change the password and port number in these examples to match the ones in your configuration environment.

1. Consult your database documentation to determine the JDBC class name and connection string for your BA Repository database.
2. Go to the `biserver-ee/tomcat/webapps/pentaho/META-INF` directory and open the `context.xml` file with any file editor.
3. Comment out the resource references that refer to databases other than Oracle, such as MySQL, MS SQL Server, and PostgreSQL. Then, add the following code to the file if it does not already exist. Be sure to 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="pentaho_operations_mart" username="pentaho_operations_
mart" 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="pentaho_operations_mart" username="pentaho_operations_
mart" maxWait="10000" maxIdle="5" maxActive="20" factory="org.apache.commons.
dbcp.BasicDataSourceFactory" type="javax.sql.DataSource" auth="Container"
name="jdbc/PDI_Operations_Mart"/>
```

4. Make sure that the `validationQuery` variable for your database is set to this: `validationQuery="select 1 from dual"`
5. Save the `context.xml` file, then close it.
6. To make sure that the changes that you made in the context.xml file take effect when Tomcat is started, navigate to the `tomcat/conf/Catalina` 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.

Next Steps

Now it is time to start your server.

Use MS SQL Server as Your Repository Database

Before you prepare your Business Analytics (BA) Repository, complete the tasks in Prepare Environment.

The BA Repository resides on the database that you installed during the Prepare Environment step, and consists of three repositories: *Jackrabbit*, *Quartz*, and *Hibernate.*

- *Jackrabbit* contains the solution repository, 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.

Initialize MS SQL Server BA Repository Database

To initialize MS SQL Server so that it serves as the BA Repository, you will need to run a few SQL scripts to create the Hibernate, Quartz, and Jackrabbit (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.

The next few sections take you through the steps to initialize the MS SQL Server BA repository database.

Adjust MS SQL Server Configuration Settings

Configure the following MS SQL Server settings in Microsoft SQL Server Management Studio or other tool of your choice.

- Select **SQL Server and Windows Authentication Mode** to use mixed authentication.
- Enable TCP/IP for MS SQL Server.
- Make sure that MS SQL Server is listening on an external IP, and not localhost.

Change Default Passwords

We recommend that you change the default passwords in the SQL script files. If you are evaluating Pentaho, you might want to skip this step.

If you do decide to make the databases more secure, use any text editor to change the passwords in these files:

- `pentaho/server/biserver-ee/data/sqlserver/create_jcr_sqlServer.sql`
- `pentaho/server/biserver-ee/data/sqlserver/create_quartz_sqlServer.sql`
- `pentaho/server/biserver-ee/data/sqlserver/create_repository_sqlServer.sql`

Run SQL Scripts

Once you change the passwords, you will need to run these SQL scripts. You will need administrator permissions on the server in order to run these scripts. The process for running SQL scripts is the same for Windows or Linux machines. The list of SQL scripts is shown in the table below.

CAUTION:

If you have a different port or different password, make sure that you change the password and port number in these examples to match the ones in your configuration.

Run the scripts from the sqlcmd utility window or from Microsoft SQL Server Management Studio.

Action	SQL Script
Create Quartz	`-i <filepath to DDL>/create_quartz_sqlServer.sql`
Create Hibernate repository	`-i <filepath to DDL>/create_repository_sqlServer.sql`
Create Jackrabbit	`-i <filepath to DDL>/create_jcr_sqlServer.sql`

Verify MS SQL Server Initialization

After you run the scripts, this list will help you verify that databases, users, and logins have been created.

1. Open MS SQL Server Management Studio.
2. In the **Object Explorer** section of the window, make sure that the Quartz, Jackrabbit (JCR), and Hibernate databases are present.
3. Navigate to Security > Logins and make sure that the appropriate users have been created.
4. Exit from MS SQL Server Management Studio tool.

Configure MS SQL Server BA Repository Database

Now that you have initialized your repository database, you will need to configure Quartz, Hibernate, and Jackrabbit for a MS SQL Server database.

By default, the examples in this section are for a MS SQL Server database that runs on port 1433. The default password is also in these examples.

CAUTION:

If you have a different port or different password, make sure that you change the password and port number in these examples to match the ones in your configuration.

Set Up Quartz on MS SQL Server BA Repository Database

Event information, such as scheduled reports, is stored in the Quartz `JobStore`. During the installation process, you must indicate where the `JobStore` is located, by modifying the `quartz.properties` file.

1. Open the `pentaho/server/biserver-ee/pentaho-solutions/system/quartz/quartz.properties` file in any text editor.
2. Locate the `#_replace_jobstore_properties` section and set the `org.quartz.jobStore.driverDelegateClass` as shown here.

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

3. Locate the `# Configure Datasources` section and 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.

Set Hibernate Settings for MS SQL Server

Modify the hibernate settings file to specify where Pentaho should find the BA Repository's hibernate config file.

NOTE:

The hibernate config file specifies driver and connection information, as well as dialects and how to handle connection closes and timeouts.

The files in this section are located in the `pentaho/server/biserver-ee/pentaho-solutions/system/hibernate` directory.

1. Open the `hibernate-settings.xml` file in a text editor. Find the `<config-file>` tags and change `postgresql.hibernate.cfg.xml` to `sqlserver.hibernate.cfg.xml` as shown.
From:

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

To:

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

2. Save and close the file.
3. Open the `sqlserver.hibernate.cfg.xml` file in a text editor.
4. Make sure that the password and port number match the ones you specified in your configuration. Make changes if necessary, then save and close the file.

Replace Default Version of Audit Log File with MS SQL Server Version

Since you are using MS SQL Server to host the BA Repository, you need to replace the `audit_sql.xml` file with one that is configured for MS SQL Server.

1. Locate the `pentaho-solutions/system/dialects/sqlserver/audit_sql.xml` file.
2. Copy it into the `pentaho-solutions/system` directory.

Modify Jackrabbit Repository Information for MS SQL Server

There are parts of code that you will need to alter in order to change the default JCR repository to MS SQL Server.

1. Navigate to the `pentaho/server/biserver-ee/pentaho-solutions/system/jackrabbit` and open the `repository.xml` file with any text editor.
2. Following the table below, locate and change the code so that the MS SQL Server lines are **not** commented out, but the MySQL, PostgreSQL and Oracle lines **are** commented out.

CAUTION:

If you have a different port or different password, make sure that you change the password and port number in these examples to match the ones in your configuration.

Item:	Code Section:

Repository	`<FileSystem class="org.apache.jackrabbit.core.fs.db.MSSqlFileSystem">`
	`<param name="driver" value="com.microsoft.sqlserver.jdbc.SQLServerDriver"/>`
	`<param name="url" value="jdbc:sqlserver://localhost:1433;DatabaseName=jackrabbit"/>`
	…
	`<param name="schema" value="mssql"/>`
	`</FileSystem>`

DataStore	`<DataStore class="org.apache.jackrabbit.core.data.db.DbDataStore">`
	`<param name="url" value="jdbc:sqlserver://localhost:1433;DatabaseName=jackrabbit"/>`
	…
	`<param name="schema" value="mssql"/>`
	`</DataStore>`

Workspaces	`<FileSystem class="org.apache.jackrabbit.core.fs.db.MSSqlFileSystem">`
	`<param name="driver" value="com.microsoft.sqlserver.jdbc.SQLServerDriver"/>`
	`<param name="url" value="jdbc:sqlserver://localhost:1433;DatabaseName=jackrabbit"/>`

Item:	Code Section:
	… <param name="schema" value="mssql"/> </FileSystem>
PersistenceManager (1st part)	<PersistenceManager class="org.apache.jackrabbit.core.persistence.bundle.MSSqlPersistenceManager"> <param name="url" value="jdbc:sqlserver://localhost:1433;DatabaseName=jackrabbit"/> … <param name="schema" value="mssql"/> </PersistenceManager>
Versioning	<FileSystem class="org.apache.jackrabbit.core.fs.db.MSSqlFileSystem"> <param name="driver" value="com.microsoft.sqlserver.jdbc.SQLServerDriver"/> <param name="url" value="jdbc:sqlserver://localhost:1433;DatabaseName=jackrabbit"/> … <param name="schema" value="mssql"/> </FileSystem>
PersistenceManager (2nd part)	<PersistenceManager class="org.apache.jackrabbit.core.persistence.bundle.MSSqlPersistenceManager"> <param name="url" value="jdbc:sqlserver://localhost:1433;DatabaseName=jackrabbit"/> … <param name="schema" value="mssql"/> </PersistenceManager>

Are you using JBoss?

If you are using JBoss, you can skip the Tomcat section and move on to Prepare JBoss Connections and Web App Servers.

Perform Tomcat-Specific Connection Tasks

After your repository has been configured, you must configure the web application servers to connect to the BA Repository. In this section, JDBC and JNDI connections are made to the Hibernate, Jackrabbit, and Quartz databases.

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.

The next couple of sections guide you through the process of working with the JDBC drivers and connection information for Tomcat.

Download Drivers and Install with the JDBC Distribution Tool

To connect to a database, including the BA Repository or DI Repository database, you will need to download and install a JDBC driver to the appropriate places for Pentaho components as well as on the the web application server that contains the Pentaho Server(s). Fortunately, the JDBC Distribution Tool makes this process easy.

NOTE:
Due to licensing restrictions, Pentaho cannot redistribute some third-party database drivers. This is why you have to download the file yourself and install it yourself.

1. Download a JDBC driver JAR from your database vendor or a third-party driver developer.
2. Copy the JDBC driver JAR you just downloaded to the `pentaho/jdbc-distribution` directory.
3. Open a cmd prompt or shell tool, navigate to the `pentaho/jdbc-distribution` directory and enter one of the following:

Windows:

```
distribute-files.bat <name of JDBC driver JAR>
```

Linux:

```
./ distribute-drivers.sh
```

1. If you have run this utility as part of the installation process, you are done. Go to the next step of the installation instructions.
2. If you have run this utility so that you can connect to a new repository, restart the BA or DI Server and Design tools, then try to connect to the new repository. If you cannot connect, verify that the drivers are installed as shown in this table. Restart your Pentaho Server(s) and Client tools.

List of Products and Corresponding Locations for JDBC Drivers

Server or Design Tool	Directory
Business Analytics (BA) Server	pentaho/server/biserver-ee/tomcat/lib
Data Integration (DI) Server	pentaho/server/data-integration-server/tomcat/lib

Server or Design Tool	Directory
Pentaho Data Integration (Spoon)	pentaho/design-tools/data-integration/lib
Pentaho Report Designer (PRD)	pentaho/design-tools/report-designer/lib/jdbc
Pentaho Aggregation Designer (PAD)	pentaho/design-tools/aggregation-designer/drivers
Pentaho Schema Workbench (PSW)	pentaho/design-tools/schema-workbench/drivers
Pentaho Metadata Editor (PME)	pentaho/design-tools/metadata-editor/libext/JDBC

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.

CAUTION:

If you have a different user or password, make sure that you change the user and password in these examples to match the ones in your configuration environment.

1. Consult your database documentation to determine the JDBC class name and connection string for your BA Repository database.
2. Go to the `biserver-ee/tomcat/webapps/pentaho/META-INF` directory and open the `context.xml` file with any file editor.
3. Comment out the resource references that refer to databases other than MS SQL Server, such as PostgreSQL, MySQL, and Oracle. Then, add the following code to the file if it does not already exist. Be sure to adjust the port numbers and passwords to reflect your environment, if necessary.

```
<Resource validationQuery="select 1"
url="jdbc:sqlserver://localhost:1433;DatabaseName=hibernate"
driverClassName="com.microsoft.sqlserver.jdbc.SQLServerDriver"
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:sqlserver://localhost:1433;DatabaseName=hibernate"
driverClassName="com.microsoft.sqlserver.jdbc.SQLServerDriver"
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:sqlserver://localhost:1433;DatabaseName=quartz" driverClassName="com.
microsoft.sqlserver.jdbc.SQLServerDriver" 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:sqlserver://localhost:1433;DatabaseName=hibernate"
driverClassName="com.microsoft.sqlserver.jdbc.SQLServerDriver"
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"/>
```

4. Modify the username, password, driver class information, IP address (or domain name), and port numbers so they reflect the correct values for your environment.
5. Make sure that the `validationQuery` variable for your database is set to this: `validationQuery="select 1"`
6. Save the `context.xml` file, then close it.
7. To make sure that the changes that you made in the context.xml file take effect when Tomcat is started, navigate to the `tomcat/conf/Catalina` 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.

Next Steps

Now it is time to start your server.

Prepare JBoss Connections and Web App Servers

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. By default, the BA Server software is configured to be deployed and run on the Tomcat server.

If you are using JBoss, both JDBC and JNDI connection information must be specified.

Perform JBoss-Specific Connection Tasks

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.

The JDBC Drivers Reference has a list of supported drivers.

Create Module File for BA Repository 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.

1. Locate 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`
   - **MS SQL Server:** `sqlserver/main`
2. 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).
3. In the `postgresql/main`, `mysql/main`, or `oracle/main` (which ever one you created), do the following things.
   A. Use an editor to create a text file named `module.xml`.
   B. Copy the appropriate code into the `module.xml` file, then modify it so that the name of the JDBC driver you just downloaded appears.
   C. Save and close the module.xml file.

Repository Type	Module code
PostgreSQL	<?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>
MySQL	<?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>
Oracle	<?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>

Repository Type	Module code
MS SQL Server	<?xml version="1.0" encoding="UTF-8"?> <module xmlns="urn:jboss:module:1.0" name="org.sqlserver"> <resources> <resource-root path="[Name of JDBC Jar You Downloaded Here]"/> </resources> <dependencies><module name="javax. api"/></dependencies> </module>

Create Module File for HSQL Database

You need to create a module file for the HSQL database.

1. Locate the `pentaho/server/biserver-ee/<your jboss installation directory>/modules/system/layers/base/org` directory and create the following path: `hsqldb/main`.
2. Download the supported JDBC driver for HSQLDB and place it in the `hsqldb/main` directory.
3. In the `hsqldb/main` directory, create a text file named `module.xml`.
4. Copy this code into 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="org.hsqldb">
<resources>
<resource-root path="[Name of JDBC Jar You Downloaded Here]"/>
</resources>
<dependencies><module name="javax.api"/></dependencies>
</module>
```

5. 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. This section shows you how to define your JNDI database connection information.

CAUTION:

If you have a different database than PostgreSQL, or if you are using a different port, password, user, driver class information, or IP address, make sure that you adjust the examples in this section to match the ones in your environment.

1. Copy the `pentaho-style.war` and `pentaho.war` files into the `pentaho/server/biserver-ee/<your jboss installation directory>/standalone/deployment` directory.
2. Locate the `pentaho/server/biserver-ee/<your jboss installation directory>/standalone/configuration/standalone.xml` file and open it with a text editor.
3. Insert these lines after the definition for `ExampleDS` data source.

```
<datasource jndi-name="java:/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:/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:/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:/PDI_Operations_Mart" pool-name="PDI_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>
                hibuser
            </user-name>
            <password>
               password
            </password>
        </security>
</datasource>
```

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-deployment-structure.xml`) to the pentaho.war.

CAUTION:

If you have a different database than PostgreSQL, adjust the module name information in this section to match the ones in your environment.

1. Use a text editor to create a new file named `jboss-deployment-structure.xml`.
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="org.postgresql" />
<module name="org.hsqldb" />
</dependencies>
</deployment>
</jboss-deployment-structure>
```

1. Save and close the file.
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 add the `jboss-deployment-structure.xml` file that you just created to it.
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.

Prepare JBoss Web Application Servers

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 `480`. 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="480"/>
```

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 by modifying the port number to match your SSL-enabled port number.

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

5. Save 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`
2. Change the following code from `-Xms1303m -Xmx1303m -XX:MaxPermSize=256m` to this:

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

3. Save the changes and close the file.

Optional: Add JBoss Logging

You can add Pentaho application level logging to the JBoss logging subsystem.

1. Open the `standalone.xml` file in the `<your jboss directory>/standalone/configuration` directory.
2. Under the `</extensions>` tag add this code.

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

3. 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>
```

4. 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"/>
```

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

```
<logger category="org.hibernate" 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>
```

6. Save and close the file.

Start BA Server

How you start the BA Server depends on your operating system.

- Windows
- Linux or Mac

Starting BA Server on Windows

Modify Tomcat Windows 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.

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 biserver-ee directory.
3. Add the java option pentaho.installed.licenses.file to `CATALINA_OPTS`. You need to modify setting of `CATALINA_OPTS` variable by adding the java option. See the following example.

```
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%
```

1. 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.

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.
   - **Windows JBoss:** Launch the `standalone.bat` file. The startup.bat file is in the JBoss `bin` directory.

2. 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.

Starting BA Server on Linux

Modify the Tomcat Linux 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.

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 biserver-ee directory.
3. Add the java option pentaho.installed.licenses.file to CATALINA_OPTS. You need to modify setting of CATALINA_OPTS variable at the end of the file by adding the java option. See the following example.

```
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"
```

1. 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.
   - **Linux Tomcat:** Launch the `startup.sh` file. The file is in the Tomcat `bin` directory.
   - **Linux JBoss:** Launch the `standalone.sh` file. The file is in the JBoss `bin` directory.
2. 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:

f 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