



Install DI Server with Your Own DI Repository

Copyright Page

This document supports Pentaho Business Analytics Suite 5.4 GA and Pentaho Data Integration 5.4 GA, documentation revision June 9th, 2015, copyright © 2015 Pentaho Corporation. No part may be reprinted without written permission from Pentaho Corporation. All trademarks are the property of their respective owners.

Help and Support Resources

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

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

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

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

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

This section explains how to install Data Integration (DI) Server and configure it to use the DI Repository database of your choice. The DI Repository contains solution content, scheduling, and audit tables needed for the DI Server to operate. You can house the DI Repository on PostgreSQL, MySQL, MS SQL Server, or Oracle. With this installation option, you must supply, install, and configure your chosen database yourself.

Prerequisites

Read [Select DI Installation Options](#) to make sure that this is the best installation option for you. Then, check the [Supported Technologies](#) tables to make sure that your server computer, DI Repository database, and web browser meet Pentaho's requirements for this version of the software.

Expertise

The topics in this section are written for IT administrators, evaluators, and analysts who have who have access to the server on which the DI Server will be installed. You should 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.

Tools

You will need a text editor and a zip tool to complete some of the steps in this installation process.

Login Credentials

All of the tasks in this section require that you have the appropriate permissions and accesses required to install software on servers and workstations.

Overview of the Installation Process

The installation process consists of the following steps.

- **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 DI Repository. Also provides information about how to configure the DI Repositories on your selected database.
- **Start DI Server:** Explains how to modify startup files and start the DI server.
- **Next Steps:** Indicates what to do after the DI Server has been installed.

Prepare Your 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

Install the DI Repository Host Database

The DI Repository houses data needed for Pentaho tools to provide scheduling and security functions. It also stores transformations and jobs. You can host the DI Repository on these databases.

- PostgreSQL
- MySQL
- Oracle
- MS SQL Server

To install the DI 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.

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.

Download and Unpack Installation Files

The Pentaho DI Server software, data files, and examples are stored in pre-packaged .zip files. You will need to manually copy these files to correct directories.

1. Download the following installation file from the [Pentaho Customer Support Portal](#) in the archive build folder.

- DI Server Installation File: `pdi-ee-server-<current version>-dist.zip`

1. Unzip the DI 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**.

7. Navigate to the `pentaho` directory and create a server subdirectory.
8. Move the `data-integration-server` directory into the `server` directory. When you are finished, the directory structure should look like this:

- `pentaho\jdbc-distribution`
- `pentaho\license-installer`
- `pentaho\server\data-integration-server`

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 DI 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 the DI Repository Host Database

The DI 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 DI Repository on these databases.

- PostgreSQL
- MySQL
- Oracle
- MS SQL Server

To install the DI 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 DI Repository database is installed correctly.

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.

Download and Unpack Installation Files

The Pentaho DI Server software, data files, and examples are stored in pre-packaged .zip files. You will need to manually copy these files to correct directories.

1. Download the following installation and plug-in files from the [Pentaho Customer Support Portal](#) in the archive build folder.

- DI Server Installation File: `pdi-ee-server-<current version>-dist.zip`.

1. Unzip the DI 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** window 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**.
5. Navigate to the `pentaho` directory and create a `server` subdirectory.
6. Move the data-integration-server directory into the server directory. When you are finished, the directory structure should look like this:

- `pentaho/jdbc-distribution`
- `pentaho/license-installer`
- `pentaho/server/data-integration-server`

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 DI 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 DI Server starts, you will need to modify the startup scripts for either the DI 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 DI 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 DI 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 DI Repository, complete the tasks in [Prepare Environment](#).

The DI 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 DI Repository Database

To initialize PostgreSQL so that it serves as the DI 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 DI 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/data-integration-server/data/postgresql/create_jcr_postgresql.sql`
- `pentaho/server/data-integration-server/data/postgresql/create_quartz_postgresql.sql`
- `pentaho/server/data-integration-server/data/postgresql/create_repository_postgresql.sql`
- `pentaho/server/data-integration-server/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** in the **pgAdminIII** tool.

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

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 DI 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 DI 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/data-integration-server/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 DI 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/data-integration-server/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/data-integration-server/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, 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	<pre> <FileSystem class="org.apache.jackrabbit.core.fs.db. DbFileSystem"> <param name="driver" value="org.postgresql.Driver"/> <param name="url" value="jdbc:postgresql://localhost:5432/di_jackrabbit"/> ... </FileSystem> </pre>
DataStore	<pre> <DataStore class="org.apache.jackrabbit.core.data.db. DbDataStore"> <param name="url" value="jdbc:postgresql://localhost:5432/di_jackrabbit"/> ... </DataStore> </pre>
Workspaces	<pre> <FileSystem class="org.apache.jackrabbit.core.fs.db. DbFileSystem"> <param name="driver" value="org.postgresql. Driver"/> <param name="url" value="jdbc:postgresql://localhost:5432/di_jackrabbit"/> ... </FileSystem> </pre>
PersistenceManager (1st part)	<pre> <PersistenceManager class="org.apache.jackrabbit.core. persistence.bundle.PostgreSQLPersistenceManager"> <param name="url" value="jdbc:postgresql://localhost:5432/di_jackrabbit"/> ... </PersistenceManager> </pre>
Versioning	<pre> <FileSystem class="org.apache.jackrabbit.core.fs.db. DbFileSystem"> </pre>

Item:	Code Section:
	<pre> <param name="driver" value="org.postgresql. Driver"/> <param name="url" value="jdbc:postgresql://localhost:5432/di_jackrabbit"/> ... </FileSystem> </pre>
PersistenceManager (2nd part)	<pre> <PersistenceManager class="org.apache.jackrabbit.core. persistence.bundle.PostgreSQLPersistenceManager"> <param name="url" value="jdbc:postgresql://localhost:5432/di_jackrabbit"/> ... </PersistenceManager> </pre>
DatabaseJournal	<pre> <Journal class="org.apache.jackrabbit.core.journal. DatabaseJournal"> <param name="revision" value="\${rep.home}/revision" /> <param name="url" value="jdbc:postgresql://localhost:5432/di_jackrabbit"/> <param name="driver" value="org.postgresql.Driver"/> <param name="user" value="jcr_user"/> <param name="password" value="password"/> <param name="schema" value="postgresql"/> <param name="schemaObjectPrefix" value="cl_j_"/> </Journal> </pre>

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.

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 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
Data Integration (DI) Server	pentaho/server/data-integration-server/tomcat/lib
Pentaho Data Integration (Spoon)	pentaho/design-tools/data-integration/lib

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 DI 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 DI 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 DI Repository database.
2. Go to the `data-integration-server/tomcat/webapps/pentaho-di/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/di_
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/di_
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/di_
quartz" driverClassName="org.postgresql.Driver" password="password"
username="pentaho_user" maxWait="10000" maxIdle="5" maxActive="20"
factory="org.apache.commons.dbcp.BasicDataSourceFactory" type="javax.sql.
DataSource" auth="Container" name="jdbc/Quartz"/>
<Resource validationQuery="select 1" url="jdbc:postgresql://localhost:5432/
hibernate" driverClassName="org.postgresql.Driver" password="password"
username="hibuser" maxWait="10000" maxIdle="5" maxActive="20" factory="org.
apache.commons.dbcp.BasicDataSourceFactory" type="javax.sql.DataSource"
auth="Container" name="jdbc/pentaho_operations_mart"/>
<Resource validationQuery="select 1" url="jdbc:postgresql://localhost:5432/
hibernate" driverClassName="org.postgresql.Driver" password="password"
username="hibuser" maxWait="10000" maxIdle="5" maxActive="20" factory="org.
```

```
apache.commons.dbcp.BasicDataSourceFactory" type="javax.sql.DataSource"
auth="Container" name="jdbc/PDI_Operations_Mart"/>
```

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-di.xml` file is in the present, delete it. It will be generated again when you start the DI 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.

- [Start the DI Server](#)

Use MySQL as Your Repository Database

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

The DI 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 DI Repository Database

To initialize MySQL so that it serves as the DI 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 DI 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/data-integration-server/data/mysql5/create_jcr_mysql.sql`
- `pentaho/server/data-integration-server/data/mysql5/create_quartz_mysql.sql`
- `pentaho/server/data-integration-server/data/mysql5/create_repository_mysql.sql`
- `pentaho/server/data-integration-server/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. Make sure that the Quartz, Jackrabbit (JCR), Hibernate, and Pentaho Operations Mart databases are present.
3. Exit from the **MySQL Workbench**.

Configure MySQL DI 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 DI 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/data-integration-server/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 DI 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/data-integration-server/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 DI 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/data-integration-server/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	<pre> <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/di_jackrabbit"/> ... </FileSystem> </pre>
DataStore	<pre> <DataStore class="org.apache.jackrabbit.core.data.db. DbDataStore"> <param name="url" value="jdbc:mysql://localhost:3306/ di_jackrabbit"/> ... </DataStore> </pre>
Workspaces	<pre> <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/di_jackrabbit"/> ... </FileSystem> </pre>
PersistenceManager (1st part)	<pre> <PersistenceManager class="org.apache.jackrabbit.core. persistence.bundle.MySqlPersistenceManager"> <param name="url" value="jdbc:mysql://localhost:3306/di_jackrabbit"/> </pre>

Item:	Code Section:
	<pre>... </PersistenceManager></pre>
Versioning	<pre><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/di_jackrabbit"/> ... </FileSystem></pre>
PersistenceManager (2nd part)	<pre><PersistenceManager class="org.apache.jackrabbit.core. persistence.bundle.MySqlPersistenceManager"> <param name="url" value="jdbc:mysql://localhost:3306/di_jackrabbit"/> ... </PersistenceManager></pre>
DatabaseJournal	<pre><Journal class="org.apache.jackrabbit.core.journal. DatabaseJournal"> <param name="revision" value="\${rep.home}/revision"/> <param name="driver" value="com.mysql.jdbc.Driver"/> <param name="url" value="jdbc:mysql://localhost:3306/ di_jackrabbit"/> <param name="user" value="jcr_user"/> <param name="password" value="password"/> <param name="schema" value="mysql"/> <param name="schemaObjectPrefix" value="J_C_"/></pre>

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 section, 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.

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
Data Integration (DI) Server	pentaho/server/data-integration-server/tomcat/lib
Pentaho Data Integration (Spoon)	pentaho/design-tools/data-integration/lib

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 DI 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 DI 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 DI Repository database.
2. Go to the `server/data-integration-server/tomcat/webapps/pentaho-di/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/di_
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/di_
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/di_
quartz" driverClassName="com.mysql.jdbc.Driver" password="password"
username="pentaho_user" maxWait="10000" maxIdle="5" maxActive="20"
factory="org.apache.commons.dbcp.BasicDataSourceFactory" type="javax.sql.
DataSource" auth="Container" name="jdbc/Quartz"/>
<Resource validationQuery="select 1" url="jdbc:mysql://localhost:3306/pentaho_
operations_mart" driverClassName="com.mysql.jdbc.Driver" password="password"
username="hibuser" maxWait="10000" maxIdle="5" maxActive="20" factory="org.
apache.commons.dbcp.BasicDataSourceFactory" type="javax.sql.DataSource"
auth="Container" name="jdbc/pentaho_operations_mart"/>
<Resource validationQuery="select 1" url="jdbc:mysql://localhost:3306/pentaho_
operations_mart" driverClassName="com.mysql.jdbc.Driver" password="password"
username="hibuser" maxWait="10000" maxIdle="5" maxActive="20" factory="org.
```

```
apache.commons.dbcp.BasicDataSourceFactory" type="javax.sql.DataSource"  
auth="Container" name="jdbc/PDI_Operations_Mart"/>
```

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-di.xml` file is in the present, delete it. It will be generated again when you start the DI 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.

- [Start the DI Server](#)

Use Oracle as Your Repository Database

Before you prepare your DI Repository, complete the tasks in [Prepare Environment](#).

The DI 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 DI Repository Database

To initialize Oracle so that it serves as the DI 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 DI 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/data-integration-server/data/oracle10g/create_jcr_ora.sql`
- `pentaho/server/data-integration-server/data/oracle10g/create_quartz_ora.sql`
- `pentaho/server/data-integration-server/data/oracle10g/create_repository_ora.sql`
- `pentaho/server/data-integration-server/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 DI 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 DI 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/data-integration-server/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 DI 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/data-integration-server/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 DI 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/data-integration-server/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	<pre> <FileSystem class="org.apache.jackrabbit.core.fs.db. OracleFileSystem"> <param name="url" value="jdbc:oracle:thin:@localhost:1521/XE"/> <param name="user" value="di_jcr_user"/> <param name="password" value="password"/> ... <param name="tablespace" value="di_pentaho_ tablespace"/> </FileSystem> </pre>
DataStore	<pre> <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="di_jcr_user"/> <param name="password" value="password"/> <param name="databaseType" value="oracle"/> ... </DataStore> </pre>
Workspaces	<pre> <FileSystem class="org.apache.jackrabbit.core.fs.db. OracleFileSystem"> <param name="url" value="jdbc:oracle:thin:@localhost:1521/XE"/> <param name="user" value="di_jcr_user"/> <param name="password" value="password"/> </pre>

Item:	Code Section:
	<pre> ... <param name="tablespace" value="di_pentaho_ tablespace"/> </FileSystem> </pre>
PersistenceManager (1st part)	<pre> <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="di_jcr_user"/> <param name="password" value="password"/> <param name="schema" value="oracle"/> <param name="schemaObjectPrefix" value="{wsp. name}_pm_ws_"/> <param name="tablespace" value="di_pentaho_ tablespace"/> </PersistenceManager> </pre>
Versioning	<pre> <FileSystem class="org.apache.jackrabbit.core.fs.db. OracleFileSystem"> <param name="url" value="jdbc:oracle:thin:@localhost:1521/XE"/> <param name="user" value="di_jcr_user"/> <param name="password" value="password"/> ... <param name="tablespace" value="di_pentaho_ tablespace"/> </FileSystem> </pre>
PersistenceManager (2nd part)	<pre> <PersistenceManager class="org.apache.jackrabbit.core. persistence.bundle.OraclePersistenceManager"> <param name="url" </pre>

Item:	Code Section:
	<pre> value="jdbc:oracle:thin:@localhost:1521/XE"/> <param name="driver" value="oracle.jdbc. OracleDriver"/> <param name="user" value="di_jcr_user"/> <param name="password" value="password"/> ... <param name="tablespace" value="di_pentaho_ tablespace"/> </PersistenceManager> </pre>
DatabaseJournal	<pre> <Journal class="org.apache.jackrabbit.core.journal. OracleDatabaseJournal"> <param name="revision" value="\${rep.home}/revision. log"/> <param name="driver" value="oracle.jdbc. OracleDriver"/> <param name="url" value="jdbc:oracle:thin:@localhost:1521/XE"/> <param name="user" value="di_jcr_user"/> <param name="password" value="pentaho"/> <param name="schema" value="oracle"/> <param name="schemaObjectPrefix" value="J_C_"/> </Journal> </pre>

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.

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
Data Integration (DI) Server	pentaho/server/data-integration-server/tomcat/lib
Pentaho Data Integration (Spoon)	pentaho/design-tools/data-integration/lib

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 DI 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 DI 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 DI Repository database.
2. Go to the `server/data-integration-server/tomcat/webapps/pentaho-di/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="di_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="di_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="di_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-di.xml` file is in the present, delete it. It will be generated again when you start the DI Server, but will contain the changes that you just made in the `context.xml` file.

Configure http and https Ports on Tomcat

Since the DI Server is configured to run on 9080 (http) and 9443 (https), you need to configure Tomcat for these ports as well.

1. Use a text editor to open the `server.xml` file, which is located in `pentaho/server/data-integration-server/<your tomcat installation directory>/conf` directory.
2. Modify the connector port settings for http and https to reflect the DI Server ports (9080 and 9443).
3. Save the changes and close the file.

Next Steps

Now it is time to start your server.

- [Start the DI Server](#)

Use MS SQL Server as Your Repository Database

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

The DI 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 DI Repository Database

To initialize MS SQL Server so that it serves as the DI 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 DI 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/data-integration-server/data/sqlserver/create_jcr_sqlServer.sql`
- `pentaho/server/data-integration-server/data/sqlserver/create_quartz_sqlServer.sql`
- `pentaho/server/data-integration-server/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 DI 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 DI 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/data-integration-server/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 DI 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/data-integration-server/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 DI 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/data-integration-server/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	<pre><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=di_ jackrabbit"/> ... <param name="schema" value="mssql"/> </FileSystem></pre>
DataStore	<pre><DataStore class="org.apache.jackrabbit.core.data.db. DbDataStore"> <param name="url" value="jdbc:sqlserver://localhost:1433;DatabaseName=di_ jackrabbit"/> ... <param name="schema" value="mssql"/> </DataStore></pre>
Workspaces	<pre><FileSystem class="org.apache.jackrabbit.core.fs.db. MSSqlFileSystem"> <param name="driver" value="com.microsoft. sqlserver.jdbc.SQLServerDriver"/></pre>

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

Item:	Code Section:
	<pre> <param name="schema" value="mssql"/> </PersistenceManager> </pre>
Journal	<pre> <Journal class="org.apache.jackrabbit.core.journal. MSSqlDatabaseJournal"> <param name="revision" value="\${rep. home}/revision.log" /> <param name="url" value="jdbc:sqlserver://localhost:1433;DatabaseName=di_ jackrabbit"/> <param name="driver" value="com. microsoft.sqlserver.jdbc.SQLServerDriver"/> <param name="user" value="jcr_user"/> <param name="password" value="password"/> <param name="schema" value="mssql"/> <param name="schemaObjectPrefix" value="cl_j_"/> </Journal> </pre>

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 section, 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.

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
Data Integration (DI) Server	pentaho/server/data-integration-server/tomcat/lib
Pentaho Data Integration (Spoon)	pentaho/design-tools/data-integration/lib

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 DI 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 DI 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 DI Repository database.
2. Go to the `server/data-integration-server/tomcat/webapps/pentaho-di/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=di_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=di_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=di_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"/>
```

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-di.xml` file is in the present, delete it. It will be generated again when you start the DI 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.

- [Start the DI Server](#)

Start the DI Server

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

- [Windows](#)
- [Linux or Mac](#)

Starting DI Server on Windows

Install License Keys Using the Command Line Interface

1. Download the .lic file you want to install.
2. Copy your .lic files to the DI Server.
3. Navigate to the `license-installer` directory: `pentaho\license-installer`
4. Run `install_license.bat` with the `install` switch and the location and name of your license file as a parameter, like this:

```
install_license.bat install "C:\Users\dvader\Downloads\Pentaho DI Enterprise Edition.lic"
```

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 `start-pentaho.bat` file, which is in the `data-integration-server` 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.

Start DI Server

Run the startup script for your web application server by launching the `start-pentaho.bat` file.

Problems Starting the DI Server?

Check out our [Troubleshooting Guide](#) for help.

Next Step

[Read the next steps.](#)

Starting DI Server on Linux

Install License Keys Using the Command Line Interface

1. Download the .lic file you want to install.
2. Copy your .lic files to the DI Server.
3. Navigate to the `license-installer` directory: `pentaho/license-installer`
4. Run `install_license.sh` with the `install` switch and the location and name of your .lic file as a parameter. You can specify multiple .lic files separated by spaces. Be sure to use backslashes to escape any spaces in the path or file name, like this:

```
install_license.sh install /home/<user name>/downloads/Pentaho\ DI\ Enterprise\
Edition.lic
```

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 `start-pentaho.sh` file, which is in the `data-integration-server` 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.

```
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.

Start DI Server

Run the `start-pentaho.sh` file.

Problems Starting the DI Server?

Check out our [Troubleshooting Guide](#) for help.

Next Step

[Read the next steps.](#)

Next Steps

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

- [Install the DI design tools.](#)
- [Configure the DI Server and design tools](#) so you can install licenses, set up datasources, and choose a security method, and more.

Note: If you have installed the DI 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 DI System instructions](#).

Learn More

- [Getting Started with PDI](#)