



# Administer DI Server

## Copyright Page

---

This document supports Pentaho Business Analytics Suite 5.2 GA and Pentaho Data Integration 5.2 GA, documentation revision October 7, 2014, copyright © 2014 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](mailto: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](mailto:sales@pentaho.com)

## Introduction

---

Administration tasks include setting up data connections, importing and exporting content, creating clusters, and configuring logging and monitoring operations.

### Prerequisites

Before you begin, install and configure PDI.

### Expertise

The topics covered 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, Linux, or Microsoft OS.

### Tools

We provide a design tool, Spoon, that you can use to perform some tasks. Other tasks must be performed using the command line interface.

## Specify Data Connections for the DI Server

Pentaho Data Integration (PDI) allows you to make connections in each job and transformation through an input step. Although users can create connections themselves, it is best to set up shared connections for your users so that they can simply select the connection they need from a list. We help you download the correct drivers, choose the connection type, and then create the connection.

- [Define Native \(JDBC\) Database Connections](#)
- [Define JNDI Connections for the DI Server](#)
- [Define OCI Connections for the DI Server](#)

### JDBC Database Connections

To connect to databases, install the driver for your database, as well as define the access protocol and settings now. You can choose from these access protocols.

- **Native (JDBC):** This is a commonly used access protocol. Please see details in the [Database Access Protocol Decision Table](#) to ensure you make an informed choice.
- **JNDI:** This also is a commonly used access type. Please see details in the [Database Access Protocol Decision Table](#) to ensure you make an informed choice.
- **ODBC:** We do not support ODBC, and it is our recommendation that you use the JDBC driver instead the ODBC driver. You should only use ODBC when there is the need to connect to unsupported databases by using the generic database driver or other specific reasons. For more information, see [Why Should You Avoid ODBC?](#) on the Pentaho public wiki.
- **OCI:** If you are connecting to an Oracle database, you must first [install the appropriate OCI driver and add the OCI connection](#).

Table 1. Database Access Protocol Decision Table

If You Are Interested In	Choose Options	
	<a href="#">Native (JDBC)</a>	<a href="#">JNDI</a>
<ul style="list-style-type: none"><li>• Understanding options</li></ul>	<p>Native (JDBC) connections are the easiest way to get going quickly. You specify the connection information in Spoon. The connections are controlled by the DI Server.</p> <p>If the connection information changes, you change it in Spoon for each connection you have defined.</p>	<p>JNDI connections are maintained in the application server, offering more advanced configuration options. One typical use case is you may want to hide security credentials from administrators of the Pentaho system. You specify the connection information by editing the <code>context.xml</code> file and selecting JNDI as the access type in Spoon.</p> <p>If the connection information changes, you change the <code>context.xml</code> file.</p>

If You Are Interested In	Choose Options	
	<a href="#">Native (JDBC)</a>	<a href="#">JNDI</a>
<ul style="list-style-type: none"> <li>• Expertise needed</li> </ul>	Knowledge of the JDBC driver and options for your RDBMS	Knowledge of Tomcat or JBoss JNDI connection procedures and options
<ul style="list-style-type: none"> <li>• How much time it takes</li> </ul>	Approximately 10 minutes	Approximately 30 minutes
<ul style="list-style-type: none"> <li>• Recommendation</li> </ul>	Use for the Pentaho Trial Download, evaluating, and rapid development.	Use for production or when the work environment is distributed in a network.

## Define Native (JDBC) Database Connections

---

Once you have [chosen to use the Native \(JDBC\) access protocol](#), here are configuration and maintenance tasks you can perform.

### Add Drivers

The DI Server and workstations need the appropriate driver to connect to the database that stores your data. Your database administrator, Chief Intelligence Officer, or IT manager should be able to provide the appropriate driver. If not, you can download drivers from your database vendor's website. See the [Supported Technologies](#) to ensure that your database and its driver are supported by Pentaho.

Note: If you are using a Microsoft SQL Server (MSSQL), you might need to use an alternative, non-vendor-supported driver called JTDS. Contact [Pentaho support](#) to ensure that you are adding the correct driver.

#### Installing Drivers

Once you have the correct driver, copy it to these directories.

**DI Server:** `/pentaho/server/data-integration-server/tomcat/webapps/pentaho-di/WEB-INF/lib/` .

**Spoon:** `data-integration/lib`

You must restart Spoon for the driver to take effect.

There should be only one driver for your database in this directory. Ensure that there are no other versions of the same vendor's driver in this directory. If there are, back up the old driver files and remove them to avoid version conflicts. This is a concern when you are adding a driver for the same database type as your Pentaho solution repository. If you have any concerns about how to proceed, contact [Pentaho support](#).

When the driver files are in place [restart](#) the server.

#### Connecting to a Microsoft SQL Server Using Integrated or Windows Authentication

If you are using a Microsoft SQL Server (MSSQL), you might need to use an alternative, non-vendor-supported driver called JTDS. Contact [Pentaho support](#) to ensure that you are adding the correct driver

For Microsoft Windows, most JDBC drivers support Type 2 integrated authentication through the `integratedSecurity` connection string property. To use integrated authentication, copy the `sqljdbc_auth.dll` file to all machines and directories to which you copied the JDBC driver. You can find this file in this location.

`<installation directory>\sqljdbc_<version>\<language>\auth\`

If running:	Use the sqljdbc_auth.dll file here:
32-bit Java Virtual Machine (JVM) even if the operating system is version x64	x86 folder
64-bit JVM on a x64 processor	x64 folder
64-bit JVM on an Itanium processor	IA64 folder

## Specify Native (JDBC) Connection Information

Before you can create the connection, you must have [installed the appropriate JDBC driver](#) for your particular data.

Pentaho Data Integration (PDI) allows you to define connections to multiple databases provided by multiple database vendors (MySQL, Oracle, PostgreSQL, and many more). PDI ships with the most suitable JDBC drivers for PostgreSQL, our default database.

Note:

Pentaho recommends that you avoid using ODBC connections. The ODBC to JDBC bridge driver does not always provide an exact match and adds another level of complexity that may affect performance. The only time you may have to use ODBC is if there is no available JDBC driver. For details, this article explains "Why you should avoid ODBC." <http://wiki.pentaho.com/pages/viewpage.action?pageId=14850644>.

When you define a database connection, the connection information (for example, the user name, password, and port number) is stored in the DI Repository and is available to other users when they connect to the repository. If you are not using the DI Repository, the database connection information is stored in the XML file associated with the transformation or job.

Connections that are available for use with a transformation or job are listed under the **Database connections** node [in the View pane in Spoon](#).

You must have information about your database, such as your database type, port number, user name and password, before you define a JDBC connection. You can also set connection properties using variables. Variables provide you with the ability to access data from multiple database types using the same transformations and jobs.

Note: Make sure to use clean ANSI SQL that works on all used database types.



**Database Connection**

**General**  
Advanced  
Options  
Pooling  
Clustering

Connection Name:

Connection Type:  
 MonetDB  
 MySQL  
 Native Mondrian  
 Neoview  
 Netezza  
 OpenERP Server  
**Oracle**  
 Oracle RDB  
 Palo MOLAP Server  
 PostgreSQL  
 Remedy Action Request System  
 SAP ERP System

Access:  
**Native (JDBC)**  
 ODBC  
 OCI  
 JNDI

**Settings**  
 Host Name:   
 Database Name:   
 Tablespace for Data:   
 Tablespace for Indices:   
 Port Number:   
 User Name:   
 Password:

Test Feature List Explore

OK Cancel

1. From within **Spoon**, navigate to the **View** tab of the **Explorer** pane. Double-click on the **Database connections** folder. The **Database Connection** dialog box appears.

Section Name	What to Do
<b>Connection Name</b>	Type name that uniquely identifies your new connection
<b>Connection Type</b>	Select the type of database to which you are connecting
<b>Access</b>	Select your method of access. Available access types depend on the connecting database type.

Section Name	What to Do
<b>Host Name</b>	Type the name of the server that hosts the database to which you are connecting. Alternatively, you can specify the host by IP address.
<b>Database Name</b>	Enter the name of the database to which you are connecting. If you are using a ODBC connection, enter the Data Source Name (DSN) in this field.
<b>Port Number</b>	Enter the TCP/IP port number if it is different from the default.
<b>User name</b>	Optionally, type the user name used to connect to the database.
<b>Password</b>	Optionally, type the password used to connect to the database.

2. Click **Test**. A confirmation message displays if Spoon is able to establish a connection with the target database.
3. Click **OK** to save your entries and exit the Database Connection dialog box.
4. From within the [View tab](#), right-click on the connection and select **Share** from the list that appears. This shares the connection with your users. They will be able to select the shared connection. From within the [View tab](#), click **Explore** to open the **Database Explorer** for an existing connection. This shows you the schemas and tables inside the connection.

## Add Database-Specific Options

Add database-specific options by adding parameters to the generated URL.

1. From within the **Database Connection** dialog box, select **Options**.
2. Select the first available row in the parameter table.
3. Choose the database type and enter a valid parameter name and its corresponding value.  
Note: For more database-specific configuration help, click **Help**. A new browser opens and displays additional information about configuring the JDBC connection for the currently selected database type.
4. Click **OK** to save your entries.

## Advanced Configuration of Database Connections

The **Advanced** option in the **Database Connection** dialog box allows you to configure properties that are, for most part, associated with how SQL is generated. These options allow you to set a standard across all of your SQL tools, ETL tools and design tools. All database table names and column names are always upper case or lower case no matter what users do in the tools.

Feature	Description
<b>Supports boolean data types</b>	Instructs PDI to use native boolean data types if supported by the database.

Feature	Description
Quote all in database	Enables the databases to use a case-sensitive tablename (for example MySQL is case-sensitive on Linux but not case sensitive on Windows). If you quote the identifiers, the databases will use a case sensitive tablename.
Force all to lower case	Enables all identifiers to lower case.
Force all to upper case	Enables all identifiers to upper case.
Preferred schema name...	Enter the preferred schema name (for example, MYSCHEMA).
Enter SQL name...	Enter the SQL statement used to initialize a connection.

Pentaho has implemented a database-specific quoting system that allows you to use any name or character acceptable to the supported databases' naming conventions.

Pentaho Data Integration contains a list of reserved words for most of the supported databases. To ensure that quoting behaves correctly, Pentaho has implemented a strict separation between the schema (user/owner) of a table and the table name itself. Doing otherwise makes it impossible to quote tables or fields with one or more periods in them correctly. Placing periods in table and field names is common practice in some ERP systems (for example, fields such as "V.A.T.")

To avoid quoting-related errors, a rule stops the Pentaho Data Integration from performing quoting activity when there is a start or end quote in the table name or schema. This allows you to specify the quoting mechanism yourself.

## Define Connection Pooling

Instead of having a connection open for each individual step in a transformation, you can set up a connection pool and define options like the initial pool size, maximum pool size, and connection pool parameters. For example, you might start with a pool of ten or fifteen connections, and as you run jobs or transformations, the unused connections drop off. Pooling helps control database access, especially if you have transformations that contain many steps and that require a large number of connections. Pooling can also be implemented when your database licensing restricts the number of active concurrent connections.

This table shows descriptions of the pooling options.

Feature	Description
Enable connection pooling	Enables connection pooling
Pool Size	Sets the <i>initial</i> size of the connection pool; sets the <i>maximum</i> number of connections in the connection pool
Parameters	Allows you to define additional custom pool parameters; click <b>Restore Defaults</b> when appropriate

Feature	Description
Description	Allows you to add a description for your parameters

1. Select **Enable Connection Pooling**.
2. Type the initial pool size in the **Initial:** area and the maximum pool size in the **Maximum:** area.
3. Select the parameters you need from within the **Parameters:** area. A Description of the parameter appears in the **Description:** area when you select a check box.
4. Click **OK** to save your selections and close the **Database Connection** dialog box.

## Connect to Clusters

This option allows you to enable clustering for the database connection and create connections to the data partitions. To create a new data partition, enter a **Partition ID** and the **Host Name**, **Port**, **Database**, **User Name**, and **Password** for connecting to the partition.

## Modify Connections

This table contains information about other database-related connection tasks you can perform.

Task	Description
Edit a Connection	Right-click on the connection name and select <b>Edit</b> .
Duplicate a Connection	Right-click on the connection name and select <b>Duplicate</b> .
Copy to a Clipboard	Allows you to copy the XML defining the step to the clipboard. You can then paste this step into another transformation. Double-click on the connection name in the tree or right-click on the connection name and select <b>Copy to Clipboard</b> .
Delete a Connection	Double-click on the connection name in the tree or right-click on the connection name and select <b>Delete</b> .
SQL Editor	To execute SQL command against an existing connection, right-click on the connection name and select <a href="#">SQL Editor</a> .
Clear the Database Cache	To speed up connections Pentaho Data Integration uses a database cache. When the information in the cache no longer represents the layout of the database, right-click on the connection in the tree and select <b>Clear DB Cache....</b> This command is commonly used when databases tables have been changed, created or deleted.
Share a Connection	Rather than redefining a connection each time you create a job or transformation on your <i>local device</i> , right-click and select <b>Share</b> to share the connection information among jobs and transformations.
Exploring the Database	Double-click on the connection name in the tree or right-click on the connection name and select <a href="#">Explore</a> .

Task	Description
<b>Show dependencies</b>	Right-click a connection name and select <b>Show dependencies</b> to see all of the transformations and jobs that use this database connection.

## Define JNDI Connections for the DI Server

---

Pentaho has supplied a way of configuring a JNDI connection for "local" Pentaho Data Integration use so that you do not have an application server continuously running during the development and testing of transformations. To configure, edit the properties file called **jdbc.properties** located at `...\data-integration-server\pentaho-solutions\system\simple-jndi`.

Note: It is important that the information stored in **jdbc.properties** mirrors the content of your application server data sources.

## Define OCI Connections for the DI Server

---

Once you have [chosen to use the OCI access protocol](#), here are configuration and maintenance tasks you can perform.

### Add Drivers

The DI Server and workstations need the appropriate driver to connect to the database that stores your data. Your database administrator, Chief Intelligence Officer, or IT manager should be able to provide the appropriate driver. If not, you can download drivers from your database vendor's website. See the [Supported Technologies](#) to ensure that your database and its driver are supported by Pentaho.

Note: If you are using a Microsoft SQL Server (MSSQL), you might need to use an alternative, non-vendor-supported driver called JTDS. Contact [Pentaho support](#) to ensure that you are adding the correct driver.

#### Installing Drivers

Once you have the correct driver, copy it to these directories.

**DI Server:** /pentaho/server/data-integration-server/tomcat/webapps/pentaho-di/WEB-INF/lib/ .

**Spoon:** data-integration/lib

You must restart Spoon for the driver to take effect.

There should be only one driver for your database in this directory. Ensure that there are no other versions of the same vendor's driver in this directory. If there are, back up the old driver files and remove them to avoid version conflicts. This is a concern when you are adding a driver for the same database type as your Pentaho solution repository. If you have any concerns about how to proceed, contact [Pentaho support](#).

When the driver files are in place [restart](#) the server.

#### Connecting to a Microsoft SQL Server Using Integrated or Windows Authentication

If you are using a Microsoft SQL Server (MSSQL), you might need to use an alternative, non-vendor-supported driver called JTDS. Contact [Pentaho support](#) to ensure that you are adding the correct driver

For Microsoft Windows, most JDBC drivers support Type 2 integrated authentication through the `integratedSecurity` connection string property. To use integrated authentication, copy the `sqljdbc_auth.dll` file to all machines and directories to which you copied the JDBC driver. You can find this file in this location.

```
<installation directory>\sqljdbc_<version>\<language>\auth\
```

If running:	Use the sqljdbc_auth.dll file here:
32-bit Java Virtual Machine (JVM) even if the operating system is version x64	x86 folder
64-bit JVM on a x64 processor	x64 folder
64-bit JVM on an Itanium processor	IA64 folder

## Create OCI Connections

1. [Start the web application and DI Servers, log into the Spoon](#), then click on **Tools > Database > Explore**. The **Data Sources** dialog box appears.
2. Click the plus icon (+) on the right and select **JDBC**. The **Database Connection** dialog box appears with **General** highlighted in the left navigation pane.
3. In the **Connection Name** field, enter a name that uniquely describes this connection. The name can have spaces, but it cannot have special characters, such as #, \$, %, and alike.
4. In the **Database Type** list, select **Oracle**.
5. In the **Access** list, select **OCI**.
6. Enter **Settings** as directed by the [Oracle OCI documentation](#).
  - a. In the **SID** field, enter the Oracle system ID that uniquely identifies the database on the system.
  - b. In the **Tablespace for Data** field, enter the name of the tablespace where the data is stored.
  - c. In the **Tablespace for Indices** field, enter the name of the tablespace where the indices are stored.
  - d. Enter the **User Name** and **Password** required to access the database.
7. Click **Test**. A success message appears if the connection is established.
8. To save the connection, click **OK** twice. This connection name appears in the list of available data sources in the **Data Sources** dialog box. If you want to use Advanced, Options, or Pooling, refer to the [Oracle OCI documentation](#) to understand how to specify these settings.

## Add Database-Specific Options

Add database-specific options by adding parameters to the generated URL.

1. From within the **Database Connection** dialog box, select **Options**.
2. Select the first available row in the parameter table.
3. Choose the database type and enter a valid parameter name and its corresponding value.  
Note: For more database-specific configuration help, click **Help**. A new browser opens and displays additional information about configuring the JDBC connection for the currently selected database type.
4. Click **OK** to save your entries.

## Advanced Configuration of Database Connections

The **Advanced** option in the **Database Connection** dialog box allows you to configure properties that are, for most part, associated with how SQL is generated. These options allow you to set a standard across all of your



SQL tools, ETL tools and design tools. All database table names and column names are always upper case or lower case no matter what users do in the tools.

Feature	Description
Supports boolean data types	Instructs PDI to use native boolean data types if supported by the database.
Quote all in database	Enables the databases to use a case-sensitive tablename (for example MySQL is case-sensitive on Linux but not case sensitive on Windows). If you quote the identifiers, the databases will use a case sensitive tablename.
Force all to lower case	Enables all identifiers to lower case.
Force all to upper case	Enables all identifiers to upper case.
Preferred schema name...	Enter the preferred schema name (for example, MYSCHEMA).
Enter SQL name...	Enter the SQL statement used to initialize a connection.

Pentaho has implemented a database-specific quoting system that allows you to use any name or character acceptable to the supported databases' naming conventions.

Pentaho Data Integration contains a list of reserved words for most of the supported databases. To ensure that quoting behaves correctly, Pentaho has implemented a strict separation between the schema (user/ owner) of a table and the table name itself. Doing otherwise makes it impossible to quote tables or fields with one or more periods in them correctly. Placing periods in table and field names is common practice in some ERP systems (for example, fields such as "V.A.T.")

To avoid quoting-related errors, a rule stops the Pentaho Data Integration from performing quoting activity when there is a start or end quote in the table name or schema. This allows you to specify the quoting mechanism yourself.

## Define Connection Pooling

Instead of having a connection open for each individual step in a transformation, you can set up a connection pool and define options like the initial pool size, maximum pool size, and connection pool parameters. For example, you might start with a pool of ten or fifteen connections, and as you run jobs or transformations, the unused connections drop off. Pooling helps control database access, especially if you have transformations that contain many steps and that require a large number of connections. Pooling can also be implemented when your database licensing restricts the number of active concurrent connections.

This table shows descriptions of the pooling options.

Feature	Description
Enable connection pooling	Enables connection pooling
Pool Size	Sets the <i>initial</i> size of the connection pool; sets the <i>maximum</i> number of connections in the connection pool
Parameters	Allows you to define additional custom pool parameters; click <b>Restore Defaults</b> when appropriate
Description	Allows you to add a description for your parameters

1. Select **Enable Connection Pooling**.
2. Type the initial pool size in the **Initial:** area and the maximum pool size in the **Maximum:** area.
3. Select the parameters you need from within the **Parameters:** area. A Description of the parameter appears in the **Description:** area when you select a check box.
4. Click **OK** to save your selections and close the **Database Connection** dialog box.

## Connect to Clusters

This option allows you to enable clustering for the database connection and create connections to the data partitions. To create a new data partition, enter a **Partition ID** and the **Host Name**, **Port**, **Database**, **User Name**, and **Password** for connecting to the partition.

## Modify Connections

This table contains information about other database-related connection tasks you can perform.

Task	Description
Edit a Connection	Right-click on the connection name and select <b>Edit</b> .
Duplicate a Connection	Right-click on the connection name and select <b>Duplicate</b> .
Copy to a Clipboard	Allows you to copy the XML defining the step to the clipboard. You can then paste this step into another transformation. Double-click on the connection name in the tree or right-click on the connection name and select <b>Copy to Clipboard</b> .
Delete a Connection	Double-click on the connection name in the tree or right-click on the connection name and select <b>Delete</b> .
SQL Editor	To execute SQL command against an existing connection, right-click on the connection name and select <a href="#">SQL Editor</a> .
Clear the Database Cache	To speed up connections Pentaho Data Integration uses a database cache. When the information in the cache no longer represents the layout of the database, right-click on the connection in the tree and select <b>Clear DB Cache....</b> This command is commonly used when databases tables have been changed, created or deleted.

Task	Description
Share a Connection	Rather than redefining a connection each time you create a job or transformation on your <i>local device</i> , right-click and select <b>Share</b> to share the connection information among jobs and transformations.
Exploring the Database	Double-click on the connection name in the tree or right-click on the connection name and select <a href="#">Explore</a> .
Show dependencies	Right-click a connection name and select <b>Show dependencies</b> to see all of the transformations and jobs that use this database connection.

## Create a Connection to the DI Repository

---

Transformations, jobs, and schedules are stored in the DI Repository. Create a connection to the DI Server if you want to access these things, or use the DI Operations Mart audit feature.

1. Make sure that your DI Repository is running.
2. Select **Tools > Repository > Connect** to open the **Repository Connection** window.
3. If a connection to the DI Repository has not already been created:
  - a. Click **Add(+)**. The **Select the repository type** window appears.
  - b. Select **DI Repository:DI Repository** and click **OK**. The **Repository Configuration** window appears.
  - c. Modify the URL associated with your repository if necessary, then click the **Test** button to ensure that the URL is properly configured. If the test fails, make sure that the port number in the URL is correct. (If you installed the DI using the Wizard, the correct port should appear in the `installation-summary.txt` file. The file is in the root directory where you installed DI.)
  - d. Click **OK** to exit the **Success** dialog box.
  - e. Enter an **ID** number and **Name** for your repository.
  - f. Click **OK** to exit the **Repository Configuration** window. Your new connection appears in the list of available repositories.
4. To connect to the repository, enter the user name and password for the repository, then click **OK**.

## Import and Export PDI Content

---

You can import and export PDI content to and from a repository by using PDI's built-in functions, explained in these subsections.

Note: Among other purposes, these procedures are useful for backing up and restoring content in the solution repository. However, users, roles, permissions, and schedules will not be included in import/export operations. If you want to back up these things, you should follow the procedure in [How To Backup the Solution Repository](#) instead.

- [Import Content Into a Repository](#)
- [Export Content From the Repository](#)

## Import Content Into a Repository

---

Follow the instructions below to import the repository. You must already be logged into the repository in Spoon before you perform this task.

1. In Spoon, go to **Tools > Repository > Import Repository**.
2. Locate the export (XML) file that contains the solution repository contents.
3. Click **Open**. The **Directory Selection** dialog box appears.
4. Select the directory in which you want to import the repository.
5. Click **OK**.
6. Enter a comment, if applicable.
7. Wait for the import process to complete.
8. Click **Close**.

The full contents of the repository are now in the directory you specified.

- [Use the Import Script From the Command Line](#)

## Use the Import Script From the Command Line

---

The import script is a command line utility that pulls content into an enterprise or database repository from two kinds of files: Individual KJB or KTR files, or complete repository export XML files.

You must also declare a rules file that defines certain parameters for the data integration content you're importing. We provide a sample file called **import-rules.xml**, included with the standard Data Integration client tool distribution. It contains all of the potential rules with comments that describe what each rule does. You can either modify this file, or copy its contents to another file; regardless, you must declare the rules file as a command line parameter.

### Options

The table below defines command line options for the import script. Options are declared with a dash: - followed by the option, then the = (equals) sign and the value.

Parameter	Definition/value
rep	The name of the enterprise or database repository to import into.
user	The repository username you will use for authentication.
pass	The password for the username you specified with <b>user</b> .
dir	The directory in the repository that you want to copy the content to.
limitdir	<b>Optional.</b> A list of comma-separated source directories to include (excluding those directories not explicitly declared).
file	The path to the repository export file that you will import from.
rules	The path to the rules file, as explained above.
comment	The comment that will be set for the new revisions of the imported transformations and jobs.
replace	Set to <b>Y</b> to replace existing transformations and jobs in the repository. Default value is <b>N</b> .
coe	Continue on error, ignoring all validation errors.
version	Shows the version, revision, and build date of the PDI instance that the import script interfaces with.

```
sh import.sh -rep=PRODUCTION -user=admin -pass=12345 -dir=/ -file=import-rules.xml -rules=import-rules.xml -coe=false -replace=true -comment="New version upload from UAT"
```



## Export Content From the Repository

---

Follow the instructions below to export the repository. You must already be logged into the repository through Spoon to complete this task.

1. In Spoon, go to **Tools > Repository > Export Repository**.
2. In the **Save As** dialog box, browse to the location where you want to save the export file.
3. Type a name for your export file in the **File Name** text box..  
Note: The export file will be saved in XML format regardless of the file extension used.
4. Click **Save**.

The export file is created in the location you specified. This XML file is a concatenation of all of the data integration content you selected. It is possible to break it up into individual KTR and KJB files by hand or through a transformation.

# Set DI Version Control and Comment Tracking Options

By default, PDI tracks versions and comments for jobs, transformations, and connection information when you save them. You can turn version control and comment tracking on or off by modifying the `repository.spring.properties` text file.

## Turn Version Control Off

To turn version control off, complete these steps. If you turn version control off, comment tracking is also turned off.

1. Exit from Spoon.
2. Stop the DI Server.
3. Open the `data-integration-server/pentaho-solution/systems/repository.spring.properties` file in a text editor.
4. Set `versioningEnabled` to `false`.

```
versioningEnabled=false
```

0. 1. Save and close the file.
1. 2. Start the DI Server.
2. 3. Start Spoon.
3. 4. To verify that version control is off, connect to the DI Repository. In the **Repository Explorer** window in the **Browse** tab, click on a file. Note that although you can see the **Access Control** the label, the **Version History** tab is hidden.

## Turn Comment Tracking Off

To turn comment tracking off, complete these steps.

0. 1. Exit from Spoon.
1. 2. Stop the DI Server.
2. 3. Open the `data-integration-server/pentaho-solution/systems/repository.spring.properties` file in a text editor.
3. 4. Set `versioningEnabled` to `true` and `versionCommentsEnabled` to `false`.

```
versioningEnabled=true  
versionCommentsEnabled=false
```

3. 1. Save and close the file.
4. 2. Start the DI Server.
5. 3. Start Spoon.
6. 4. To verify that comment tracking is off, connect to the DI Repository. In the **Repository Explorer** window in the **Browse** tab, click on a file. Note that the **Version History** tab appears, but that the **Comment** field is hidden. Also note that when you save a transformation, job, or connection information, you are no longer prompted to enter a comment.

## Turn Version Control On

To turn version control on, do these things.

1. Exit from Spoon.
2. Stop the DI Server.
3. Open the `data-integration-server/pentaho-solution/systems/repository.spring.properties` file in a text editor.
4. Set `versioningEnabled` to `true`.

```
versioningEnabled=true
```

1. If you want to also turn Comment Tracking, make sure to set `versionCommentsEnabled` to `true`.
2. Save and close the file.
3. Start the DI Server.
4. Start Spoon.
5. To verify that Version Control is on, connect to the DI Repository. In the **Repository Explorer** window in the **Browse** tab, click on a file. Note that the **Version History** tab appears. When you save a transformation, job, or connection information, version control is applied.

## Turn Comment Tracking On

To turn comment tracking on, complete these steps.

1. Exit from Spoon.
2. Stop the DI Server.
3. Open the `data-integration-server/pentaho-solution/systems/repository.spring.properties` file in a text editor.
4. Set `versioningEnabled` to `true` and `versionCommentsEnabled` to `true`.

```
versioningEnabled=true  
versionCommentsEnabled=true
```

1. Save and close the file.

2. Start the DI Server.
3. Start Spoon.
4. To verify that Comment Tracking is on, connect to the DI Repository. In the **Repository Explorer** window in the **Browse** tab, click on a file. Note that the **Version History** tab appears with the **Comments** field. When you save a transformation, job, or connection information, you are prompted to enter a comment.

## Purge Content and Connection Information from the DI Repository

The Purge Utility allows you to permanently shared objects (servers, clusters, and databases) stored in the DI Repository as well as content (transformations and jobs). You can also delete revision information for content and shared objects.

### CAUTION:

Purging is permanent. Purged items cannot be restored.

To use the Purge Utility, complete these steps.

1. Make sure the DI Repository is running.
2. Open a shell tool, command prompt window, or terminal window, and navigate to the `pentaho/design-tools/data-integration` directory.
3. At the prompt enter the purge utility command. The format for the command, a table that describes each parameter, and parameter examples follow.

### NOTE:

The command must contain the `url`, `user`, and `password` parameters, as well as one of these parameters: `versionCount`, `purgeBeforeDate`, `purgeFiles`, `purgeRevisions`.

### Windows:

```
purge-utility.bat [-url] [-user] [-password] [-purgeTarget] [-versionCount]
[-purgeBeforeDate] [-purgeFiles] [-purgeRevisions] [-logFileName] [-logLevel]
```

### Linux:

```
purge-utility.sh [-url] [-user] [-password] [-purgeTarget] [-versionCount]
[-purgeBeforeDate] [-purgeFiles] [-purgeRevisions] [-logFileName] [-logLevel]
```

Option	Required?	Description
<code>-url</code>	Y	URL address for the DI Repository. This is a required parameter. By default, this the DI Server is installed at this URL: <a href="http://localhost:9080">http://localhost:9080</a> .
<code>-user</code>	Y	Username for an account that can access the DI Server as an administrator. This is a required parameter.
<code>-password</code>	Y	Password for the account used to access the DI Server. This is a required parameter.

-purgeTarget	N	Indicates whether to purge content, shared objects, or both. You can set purgeTarget to CONTENT, SHARED, BOTH, C, S, and B. (C = Content, S = Shared, and B = Both). If you do not indicate a purgeTarget, the utility will purge content by default.
-versionCount	You must include <b>only one</b> of these: versionCount, purgeBeforeDate, purgeFiles, or purgeRevisions	Deletes entire version history except the for last versionCount versions. Set this value to an integer.
-purgeBeforeDate		Deletes all versions before purgeBeforeDate. The format for the date must be mm/dd/yyyy.
-purgeFiles		When set to TRUE, content, shared object, or both types of files are permanently and physically removed. The type of file (or files) removed depends on what is set in the purgeTarget parameter. If the purgeTarget parameter is not set, content files are purged by default.
-purgeRevisions		When set to TRUE, all revisions are purged, but the current file remains unchanged.
-logFileName	N	Allows you to specify the file name for the log file. If this parameter is not present, the log is written to a file that has this name format: purge-utility-log-YYYYMMdd-HH:mm:ss.txt. YYYYMMdd-HH:mm:ss indicates the date and time that the log file was created (e.g. purge-utility-log-20140313-154741.txt).
-logLevel	N	Indicates the types and levels of detail the logs should contain. Values are: ALL, DEBUG, ERROR, FATAL, TRACE, INFO, OFF, and WARN. By default the log is set to INFO. Check the Log4j documentation for more details on the logging framework definitions: <a href="https://logging.apache.org/log4j/1.2/apidocs/org/apache/log4j/Level.html">https://logging.apache.org/log4j/1.2/apidocs/org/apache/log4j/Level.html</a> .

- In this example, only the last five revisions of transformations and jobs are NOT deleted. All previous revisions are deleted.

```
purge-utility.bat -url=http://localhost:9080 -user=jdoe -password=mypassword -versionCount=5
```

- In the example that follows all revisions before 01/11/2009 are deleted. Logging is set to the WARN level.

```
purge-utility.bat -url=http://localhost:9080 -user=jdoe -password=mypassword -purgeBeforeDate=01/11/2009 -logLevel=WARN
```

- In this example, all shared objects before 01/11/2007 are deleted. Logging is turned OFF.

```
purge-utility.bat -url=http://localhost:9080 -user=jdoe -password=mypassword  
-purgeBeforeDate=01/11/2007 -purgeTarget=S -logLevel=OFF
```

1. When finished, examine the logs to see if there were any issues or problems with the purge.
2. To see the results of the purge process, disconnect, then reconnect to the DI Repository. In the **Repository Explorer**, in the **Browse** tab, verify that the items you specified in your purge utility command were purged.

## Create Carte Clusters

---

You can set up Carte to operate as a standalone execution engine for a job or transformation. Within Spoon, you can define one or more Carte servers and send jobs and transformations to them on an individual basis. However, in some cases you will want to set up a cluster of Carte servers so that you don't have to manage Carte instance assignments by hand. You may also need to use several servers to improve performance on resource-intensive jobs or transformations. In these scenarios, you will establish a cluster of Carte servers. There are two paradigms for Carte clustering:

A **static cluster** is a Spoon instance managing Carte slave nodes that have been explicitly defined in the user interface.

A **dynamic cluster** is a single master Carte server with a variable number of available Carte slave node registered with it.

Static clusters are a good choice for smaller environments where you don't have a lot of machines (virtual or real) to use for PDI transformations. Dynamic clusters are more appropriate in environments where transformation performance is extremely important, or there can potentially be multiple concurrent transformation executions. Architecturally, the primary difference between a static and dynamic cluster is whether it's Spoon or Carte doing the load balancing.

- [Configure Carte to Be a Static Slave Instance](#)
- [Configure a Dynamic Cluster](#)
- [Create a Cluster Schema in Spoon](#)
- [Execute Transformations in a Cluster](#)
- [Initialize Slave Servers in Spoon](#)
- [Execute Scheduled Jobs on a Remote Carte Server](#)



## Configure Carte to Be a Static Slave Instance

---

Follow the directions below to set up static Carte slave servers.

Note: If you already have Carte installed on the target machines, you can skip the initial installation steps.

1. Retrieve a **pdi-ee-client** archive package from the Pentaho Customer Support Portal.
2. On each machine that will act as a Carte server (slave), create a `/pentaho/design-tools/` directory.
3. Unpack the archive to the `/pentaho/design-tools/` directory on each machine. Two directories will be created: **data-integration** and **license-installer**.
4. Use the license utility to install the PDI Enterprise Edition license, if applicable.
5. Copy over any required JDBC drivers and PDI plugins from your development instances of PDI to the Carte instances.
6. Run the Carte script with an IP address, hostname, or domain name of this server, and the port number you want it to be available on.

```
./carte.sh 127.0.0.1 8081
```

7. If you will be executing content stored in a DI Repository, copy the **repositories.xml** file from the `.kettle` directory on your workstation to the same location on your Carte slave. Without this file, the Carte slave will be unable to connect to the DI Repository to retrieve content.
8. Ensure that the Carte service is running as intended, accessible from your primary PDI development machines, and that it can run your jobs and transformations.
9. To start this slave server every time the operating system boots, create a startup or init script to run Carte at boot time with the same options you tested with.

You now have one or more Carte slave servers that you can delegate job and transformation work to in the Repository Explorer.

## Configure a Dynamic Cluster

---

Follow the procedures below to set up one or more Carte slave servers and a Carte master server to load-balance them.

- [Configure Carte as a Master \(Load Balancer\)](#)
- [Configure Carte to Be a Dynamic Slave Instance](#)

## Configure Carte as a Master (Load Balancer)

---

This procedure is only necessary for **dynamic cluster** scenarios in which one Carte server will load-balance multiple slave Carte instances. If you are implementing a static cluster, which is where Carte slaves are individually declared in the PDI user interface, then skip these instructions.

Follow the process below to establish a dynamic Carte load balancer (master server).

Note: You do not have to use Carte as a load balancer; you can use the DI Server instead. If you decide to use the DI Server, you must enable the proxy trusting filter as explained in [Execute Scheduled Jobs on a Remote Carte Server](#), then set up your dynamic Carte slaves and define the DI Server as the master.

Note: If you already have Carte installed on the target machine, you can skip the initial installation steps.

1. Retrieve a **pdi-ee-client** archive package from the Pentaho Customer Support Portal.
2. Create a `/pentaho/design-tools/` directory.
3. Unpack the archive to the `/pentaho/design-tools/` directory on each machine. Two directories will be created: **data-integration** and **license-installer**.
4. Copy over any required JDBC drivers from your development instances of PDI to the Carte instances.
5. Create a **carte-master-config.xml** configuration file using the following example as a basis:

```
<slave_config>
<!-- on a master server, the slaveserver node contains information about
this Carte instance -->
<slaveserver>
<name>Master</name>
<hostname>localhost</hostname>
<port>9001</port>
<username>cluster</username>
<password>cluster</password>
<master>Y</master>
</slaveserver>
</slave_config>
```

Note: The **<name>** must be unique among all Carte instances in the cluster.

6. Run the Carte script with the `carte-slave-config.xml` parameter.

```
./carte.sh carte-master-config.xml
```

7. Ensure that the Carte service is running as intended.
8. To start this master server every time the operating system boots, create a startup or init script to run Carte at boot time with the same config file option you specified earlier.

You now have a Carte master to use in a dynamic cluster. You must configure one or more Carte slave servers in order for this to be useful.

## Configure Carte to Be a Dynamic Slave Instance

---

Follow the directions below to set up static Carte slave servers.

Note: If you already have Carte installed on the target machines, you can skip the initial installation steps.

1. Retrieve a **pdi-ee-client** archive package from the Pentaho Customer Support Portal.
2. On each machine that will act as a Carte server (slave), create a `/pentaho/design-tools/` directory.
3. Unpack the archive to the `/pentaho/design-tools/` directory on each machine. Two directories will be created: **data-integration** and **license-installer**.
4. Copy over any required JDBC drivers and PDI plugins from your development instances of PDI to the Carte instances.
5. Create a **carte-slave-config.xml** configuration file using the following example as a basis:

```
<slave_config>
<!-- the masters node defines one or more load balancing Carte instances
that will manage this slave -->
<masters>
<slaveserver>
<name>Master</name>
<hostname>localhost</hostname>
<port>9000</port>
<!-- uncomment the next line if you want the DI Server to act as the load
balancer -->
<!-- <webAppName>pentaho-di</webAppName> -->
<username>cluster</username>
<password>cluster</password>
<master>Y</master>
</slaveserver>
</masters>

<report_to_masters>Y</report_to_masters>
<!-- the slaveserver node contains information about this Carte slave
instance -->
<slaveserver>
<name>SlaveOne</name>
<hostname>localhost</hostname>
<port>9001</port>
<username>cluster</username>
```

```
<password>cluster</password>
<master>N</master>
</slaveserver>
</slave_config>
```

Note: The slaveserver **<name>** must be unique among all Carte instances in the cluster.

6. Run the Carte script with the `carte-slave-config.xml` parameter.

```
./carte.sh carte-slave-config.xml
```

7. If you will be executing content stored in a DI Repository, copy the **repositories.xml** file from the `.kettle` directory on your workstation to the same location on your Carte slave. Without this file, the Carte slave will be unable to connect to the DI Repository to retrieve PDI content.
8. Ensure that the Carte service is running as intended.
9. To start this slave server every time the operating system boots, create a startup or init script to run Carte at boot time with the same config file option you specified earlier.

You now have a Carte slave to use in a dynamic cluster. You must configure a Carte master server or use the DI Server as a load balancer.

## Create a Cluster Schema in Spoon

---

Clustering allows transformations and transformation steps to be executed in parallel on more than one Carte server. The clustering schema defines which slave servers you want to assign to the cluster and a variety of clustered execution options.

Begin by selecting the **Kettle cluster schemas** node in the Spoon **Explorer View**. Right-click and select **New** to open the **Clustering Schema** dialog box.

Option	Description
Schema name	The name of the clustering schema
Port	Specify the port from which to start numbering ports for the slave servers. Each additional clustered step executing on a slave server will consume an additional port. Note: To avoid networking problems, make sure no other networking protocols are in the same range .
Sockets buffer size	The internal buffer size to use
Sockets flush interval rows	The number of rows after which the internal buffer is sent completely over the network and emptied.
Sockets data compressed?	When enabled, all data is compressed using the Gzip compression algorithm to minimize network traffic
Dynamic cluster	If checked, a master Carte server will perform load-balancing operations, and you must define the master as a slave server in the feild below. If unchecked, Spoon will act as the load balancer, and you must define the available Carte slaves in the field below.
Slave Servers	A list of the servers to be used in the cluster. You must have one master server and any number of slave servers. To add servers to the cluster, click <b>Select slave servers</b> to select from the list of available slave servers.

## Execute Transformations in a Cluster

---

To run a transformation on a cluster, access the **Execute a transformation** screen and select **Execute clustered**.

To run a clustered transformation via a job, access the **Transformation** job entry details screen and select the **Advanced** tab, then select **Run this transformation in a clustered mode?**.

To assign a cluster to an individual transformation step, right-click on the step and select **Clusterings** from the context menu. This will bring up the cluster schema list. Select a schema, then click **OK**.

When running transformations in a clustered environment, you have the following options:

- **Post transformation** — Splits the transformation and post it to the different master and slave servers
- **Prepare execution** — Runs the initialization phase of the transformation on the master and slave servers
- **Prepare execution** — Runs the initialization phase of the transformation on the master and slave servers
- **Start execution** — Starts the actual execution of the master and slave transformations.
- **Show transformations** — Displays the generated (converted) transformations that will be executed on the cluster

## Initialize Slave Servers in Spoon

---

Follow the instructions below to configure PDI to work with Carte slave servers.

1. Open a transformation.
2. In the **Explorer View** in Spoon, select **Slave Server**.
3. Right-click and select **New**. The **Slave Server** dialog box appears.
4. In the Slave Server dialog box, enter the appropriate connection information for the Data Integration (or Carte) slave server. The image below displays a connection to the Data Integration slave server.

Option	Description
Server name	The name of the slave server
Hostname or IP address	The address of the device to be used as a slave
Port	Defines the port you are for communicating with the remote server
Web App Name	Used for connecting to the DI server and set to pentaho-di by default
User name	Enter the user name for accessing the remote server
Password	Enter the password for accessing the remote server
Is the master	Enables this server as the master server in any clustered executions of the transformation

Note: When executing a transformation or job in a clustered environment, you should have one server set up as the master and all remaining servers in the cluster as slaves.

Below are the proxy tab options:

Option	Description
Proxy server hostname	Sets the host name for the Proxy server you are using
The proxy server port	Sets the port number used for communicating with the proxy
Ignore proxy for hosts: regexp   separated	Specify the server(s) for which the proxy should not be active. This option supports specifying multiple servers using regular expressions. You can also add multiple servers and expressions separated by the '   ' character.



5. Click **OK** to exit the dialog box. Notice that a plus sign (+) appears next to **Slave Server** in the Explorer View.

## Execute Scheduled Jobs on a Remote Carte Server

---

Follow the instructions below if you need to schedule a job to run on a remote Carte server. Without making these configuration changes, you will be unable to remotely execute scheduled jobs.

Note: This process is also required for using the DI Server as a load balancer in a dynamic Carte cluster.

1. Stop the DI Server and remote Carte server.
2. Copy the **repositories.xml** file from the `.kettle` directory on your workstation to the same location on your Carte slave. Without this file, the Carte slave will be unable to connect to the DI Repository to retrieve PDI content.
3. Open the `/pentaho/server/data-integration-server/tomcat/webapps/pentaho-di/WEB-INF/web.xml` file with a text editor.
4. Find the **Proxy Trusting Filter** filter section, and add your Carte server's IP address to the **param-value** element.

```
<filter>
  <filter-name>Proxy Trusting Filter</filter-name>
  <filter-class>org.pentaho.platform.web.http.filters.
ProxyTrustingFilter</filter-class>
  <init-param>
    <param-name>TrustedIpAddr</param-name>
    <param-value>127.0.0.1,192.168.0.1</param-value>
    <description>Comma separated list of IP addresses of a trusted hosts.
  </description>
  </init-param>
  <init-param>
    <param-name>NewSessionPerRequest</param-name>
    <param-value>true</param-value>
    <description>true to never re-use an existing IPentahoSession in the
HTTP session; needs to be true to work around code put in for BISERVER-
2639</description>
  </init-param>
</filter>
```

5. Uncomment the proxy trusting filter-mappings between the `<!-- begin trust -->` and `<!-- end trust -->` markers.

```
<!-- begin trust -->
<filter-mapping>
  <filter-name>Proxy Trusting Filter</filter-name>
```

```

    <url-pattern>/webservices/authorizationPolicy</url-pattern>
</filter-mapping>

<filter-mapping>
    <filter-name>Proxy Trusting Filter</filter-name>
    <url-pattern>/webservices/roleBindingDao</url-pattern>
</filter-mapping>

<filter-mapping>
    <filter-name>Proxy Trusting Filter</filter-name>
    <url-pattern>/webservices/userRoleListService</url-pattern>
</filter-mapping>

<filter-mapping>
    <filter-name>Proxy Trusting Filter</filter-name>
    <url-pattern>/webservices/unifiedRepository</url-pattern>
</filter-mapping>

<filter-mapping>
    <filter-name>Proxy Trusting Filter</filter-name>
    <url-pattern>/webservices/userRoleService</url-pattern>
</filter-mapping>

<filter-mapping>
    <filter-name>Proxy Trusting Filter</filter-name>
    <url-pattern>/webservices/Scheduler</url-pattern>
</filter-mapping>

<filter-mapping>
    <filter-name>Proxy Trusting Filter</filter-name>
    <url-pattern>/webservices/repositorySync</url-pattern>
</filter-mapping>
<!-- end trust -->

```

6. Save and close the file, then edit the **carte.sh** or **Carte.bat** startup script on the machine that runs your Carte server.
7. Add **-Dpentaho.repository.client.attemptTrust=true** to the **java** line at the bottom of the file.

```

java $OPT -Dpentaho.repository.client.attemptTrust=true org.pentaho.di.www.
Carte "${1+$@}"

```

8. Save and close the file.

## 9. Start your Carte and DI Server

You can now schedule a job to run on a remote Carte instance.

## Install License Keys Using the Command Line Interface

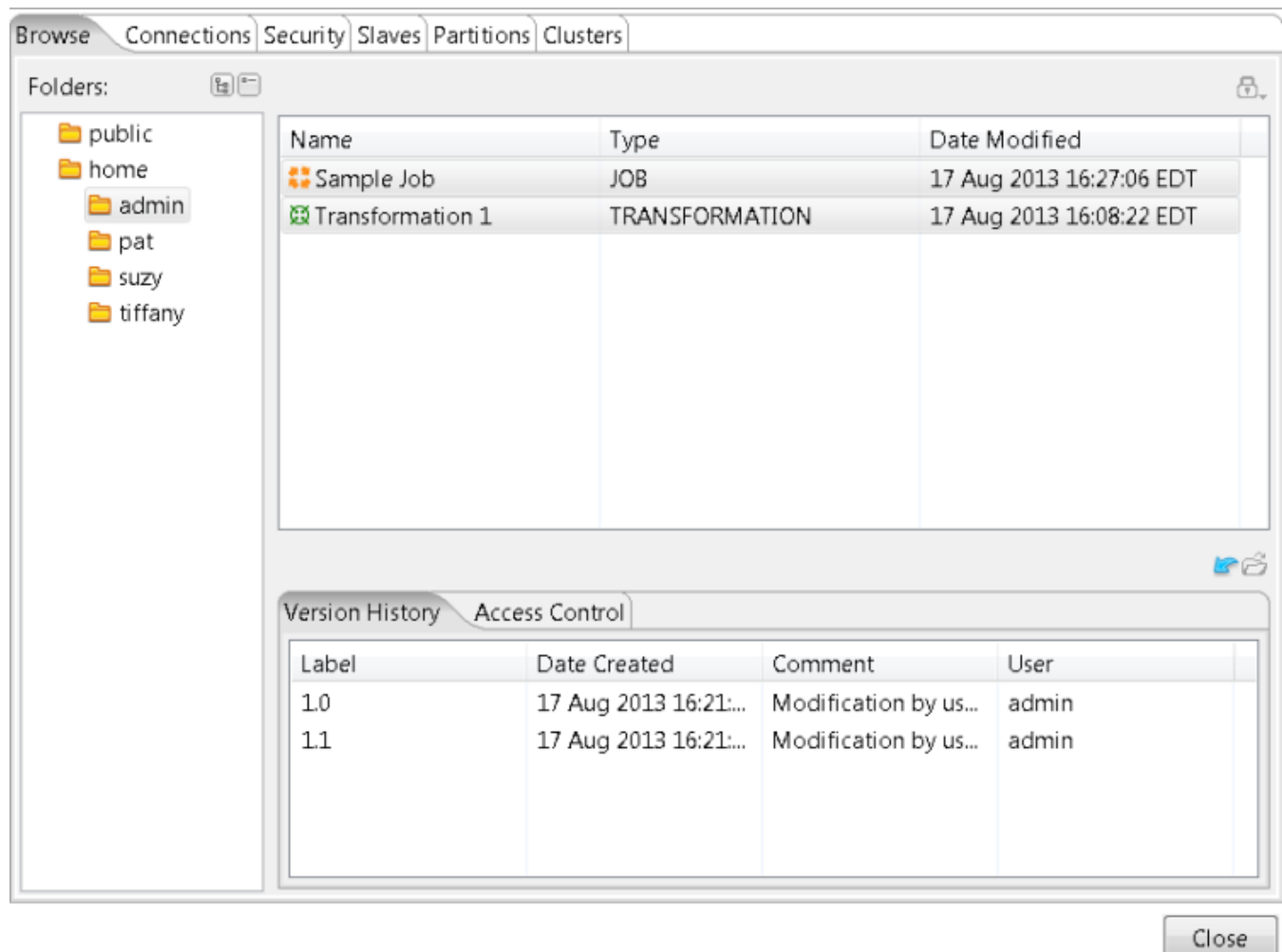
---

1. Download the .lic file you want to install.
2. Copy your .lic files to the DI Server.
3. Navigate to the `licenses` directory. `pdi/pdi-ee/data-integration/licenses`
4. Run the license installation script.
  - a. **For Linux:** 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. `install_license.sh install /home/dvader/downloads/Pentaho\ DI\ Enterprise\ Edition.lic`
  - b. **For Windows:** Run `install_license.bat` with the `install` switch and the location and name of your license file as a parameter. `install_license.bat install "C:\Users\dvader\Downloads\Pentaho DI Enterprise Edition.lic"`

## Assign Permissions to Use or Manage Database Connections

You may have several connections to your data that you do not want to share with all of your users. When connected to the DI Server, Spoon gives you the ability to make your data visible to only the users and roles that you specify. You can assign permission to allow users and roles to read, write, or delete the connection. You can also delegate the ability to assign these permissions to another user or role.

Connection definitions are stored in the DI Repository. The Spoon Repository Explorer enables you to browse the available connections and select the one for which you want to assign permissions.



1. From within **Spoon**, click on **Tools > Repository > Explore**. The **Repository Explorer on [Your\_DI\_Repository\_Name]** dialog box appears.
2. Select the **Connections** tab.
3. Select the connection for which you want to assign permissions.

4. From the User/Role area, select the user or role for which you want to assign permissions.
5. Check the permissions you want to assign to the selected user or role.

Selection	Selection Result
Read	For this user or role, the connection appears in the connection list and can be selected for use. If users or roles have permission to read a transformation or job but not to a referenced database connection, they cannot open the transformation or job and an error message appears.
Write	This user or role can edit the connection definition.
Delete	This user or role can permanently remove the connection definition from the list..
Manage Access Control	This user or role can assign read, write, and delete permissions to other users or roles.

6. Click **Apply**.
7. Click **Close** to exit the dialog box.

## List of Server Ports Used by PDI

---

The port numbers below must be available internally on the machine that runs the DI Server. The only exception is SampleData, which is only for evaluation and demonstration purposes and is not necessary for production systems. If you are unable to open these ports, or if you have port collisions with existing services, refer to [Change Service Port Numbers](#) for instructions on how to change them.

Service	Port Number
Data Integration Server	9080
H2 (SampleData)	9092
Embedded BA Server (Jetty)	10000

- [Change Service Port Numbers](#)



## Change Service Port Numbers

---

### DI Server (Tomcat)

Edit the `/pentaho/server/data-integration-server/tomcat/conf/server.xml` file and change the port numbers in the section shown below.

```
<!-- A "Connector" represents an endpoint by which requests are received
      and responses are returned. Documentation at :
      Java HTTP Connector: /docs/config/http.html (blocking & non-blocking)
      Java AJP  Connector: /docs/config/ajp.html
      APR (HTTP/AJP) Connector: /docs/apr.html
      Define a non-SSL HTTP/1.1 Connector on port 9080
-->
<Connector URIEncoding="UTF-8" port="9080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="9443" />
<!-- A "Connector" using the shared thread pool-->
<!--
<Connector URIEncoding="UTF-8" executor="tomcatThreadPool"
           port="9080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="9443" />
```

Note: You may also have to change the SSL and SHUTDOWN ports in this file, depending on your configuration. Next, follow the directions in [Change the DI Server URL](#) to accommodate for the new port number.

### Embedded BA Server (Jetty)

This server port is hard-coded in Pentaho Data Integration and cannot be changed. If port 10000 is unavailable, the system will increment by 1 until an available port is found.

## Change the DI Server URL

---

You can change the DI Server hostname from localhost to a specific IP address, hostname, or domain name by following these instructions. This procedure is also a requirement if you are changing the DI Server port number.

1. Stop the DI Server through your preferred means.
2. Open the `/pentaho/server/data-integration-server/tomcat/webapps/pentaho-di/WEB-INF/web.xml` file with a text editor.
3. Modify the value of the **fully-qualified-server-url** element appropriately.

```
<context-param>
  <param-name>fully-qualified-server-url</param-name>
  <param-value>http://localhost:9080/pentaho-di/</param-value>
</context-param>
```

4. Save and close the file.
5. Start the DI Server.

The DI Server is now configured to reference itself at the specified URL.

## Logging and Monitoring Operations

---

This section contains information on DI Server and client tool logging and status monitoring.

- [Enable Logging](#)
- [Monitor Job and Transformation Results](#)
- [Use Checkpoints to Restart Jobs](#)
- [DI Operations Mart](#)
- [Apply Best Practices for Kettle Logging](#)

## Enable Logging

---

The logging functionality in Data Integration enables you to more easily troubleshoot complex errors and failures, and measure performance. To turn on logging in Data Integration, follow the below procedure.

1. Create a database or table space called **pdi\_logging**.
2. Start Spoon, and open a transformation or job for which you want to enable logging.
3. Go to the **Edit** menu and select **Settings...** The **Settings** dialog appears.
4. Select the **Logging** tab.
5. In the list on the left, select the function you want to log.
6. Click the **New** button next to the **Log Connection** field. The **Database Connection** dialogue appears.
7. Enter your database connection details, then click **Test** to ensure that they are correct. Click **OK** when you are done.
8. Look through the list of fields to log, and ensure that the correct fields are selected.  
Warning: Monitoring the **LOG\_FIELD** field can negatively impact BA Server or DI Server performance. However, if you don't select all fields, including LOG\_FIELD, when configuring transformation logging, you will not see information about this transformation in the Operations Mart logging.

Logging is enabled for the job or transformation.

When you run a job or transformation that has logging enabled, you have the option of choosing the log verbosity level in the execution dialogue:

- **Nothing** Do not record any output
- **Error** Only show errors
- **Minimal** Only use minimal logging
- **Basic** This is the default level
- **Detailed** Give detailed logging output
- **Debug** For debugging purposes, very detailed output
- **Row level** Logging at a row level. This will generate a lot of log data

If the **Enable time** option is enabled, all lines in the logging will be preceded by the time of day.

## Log Rotation

This procedure assumes that you do not have or do not want to use an operating system-level log rotation service. If you are using such a service on your Pentaho server, connect to the BA Server and Data Integration Server and use that instead of implementing this solution.

The Business Analysis and Data Integration servers use the Apache log4j Java logging framework to store server feedback. The default settings in the log4j.xml configuration file may be too verbose and grow too large for some production environments. Follow these instructions to modify the settings so that Pentaho server log files are rotated and compressed.

1. Stop all relevant Pentaho servers.

2. Download a .zip archive of the Apache Extras Companion for log4j package: [Apache Logging Services](#).
3. Unpack the **apache-log4j-extras** JAR file from the zip archive, and copy it to the following locations:
  - **Business Analytics Server:** /tomcat/webapps/pentaho/WEB-INF/lib/
  - **Data Integration Server:** /tomcat/webapps/pentaho-di/WEB-INF/lib/
4. Edit the **log4j.xml** settings file for each server that you are configuring. The files are in the following locations:
  - **BA Server:** /tomcat/webapps/pentaho/WEB-INF/classes/
  - **DI Server:** /tomcat/webapps/pentaho-di/WEB-INF/classes/
5. Remove all **PENTAHOCONSOLE** appenders from the configuration.
6. Modify the **PENTAHOFILE** appenders to match the log rotation conditions that you prefer. You may need to consult the log4j documentation to learn more about configuration options. Two examples that many Pentaho customers find useful are listed:

#### Daily (date-based) log rotation with compression:

```
<appender name="PENTAHOFILE" class="org.apache.log4j.rolling.
RollingFileAppender">
  <!-- The active file to log to; this example is for BA/DI Server.-->
  <param name="File" value="../logs/pentaho.log" />
  <param name="Append" value="false" />
  <rollingPolicy class="org.apache.log4j.rolling.TimeBasedRollingPolicy">
    <!-- See javadoc for TimeBasedRollingPolicy -->
    <param name="FileNamePattern" value="../logs/pentaho.%d.log.gz" />
  </rollingPolicy>
  <layout class="org.apache.log4j.PatternLayout">
    <param name="ConversionPattern" value="%d %-5p [%c] %m%n"/>
  </layout>
</appender>
```

#### Size-based log rotation with compression:

```
<appender name="PENTAHOFILE" class="org.apache.log4j.rolling.
RollingFileAppender">
  <!-- The active file to log to; this example is for BA/DI Server.-->
  <param name="File" value="../logs/pentaho.log" />
  <param name="Append" value="false" />
  <rollingPolicy class="org.apache.log4j.rolling.
FixedWindowRollingPolicy">
    <param name="FileNamePattern" value="../logs/pentaho.%i.log.gz" />
    <param name="maxIndex" value="10" />
    <param name="minIndex" value="1" />
  </rollingPolicy>
  <triggeringPolicy class="org.apache.log4j.rolling.
```

```
SizeBasedTriggeringPolicy">
    <!-- size in bytes -->
    <param name="MaxFileSize" value="10000000" />
</triggeringPolicy>
<layout class="org.apache.log4j.PatternLayout">
    <param name="ConversionPattern" value="%d %-5p [%c] %m%n" />
</layout>
</appender>
```

7. Save and close the file, then start all affected servers to test the configuration.

You have an independent log rotation system in place for all modified Pentaho servers.

## Monitor Job and Transformation Results

---

You can view remotely executed and scheduled job and transformation details, including the date and time that they were run, and their status and results, through the **Kettle Status** page. To view it, navigate to the `/pentaho-di/kettle/status` page on your Data Integration Server (change the host name and port to match your configuration):

```
http://internal-di-server:9080/pentaho-di/kettle/status
```

You must be logged in to ensure you are redirected to the login page.

You can get to a similar page in Spoon by using the **Monitor** function of a slave server.

Notice the **Configuration details** table at the bottom of the screen. This shows the three configurable settings for schedule and remote execution logging. See [slave-server-config.xml](#) for more information on what these settings do and how you can modify them.

Note: This page clears when the server is restarted, or at the interval specified by the `object_timeout_minutes` setting.

- [slave-server-config.xml](#)

## slave-server-config.xml

Any action done through the Carte server embedded in the Data Integration Server is controlled through the `/pentaho/server/data-integration-server/pentaho-solutions/system/kettle/slave-server-config.xml` file. These three configurable options are explained here.

Note: To make modifications to `slave-server-config.xml`, you must stop the Data Integration Server.

Property	Values	Description
<code>max_log_lines</code>	Any value of 0 (zero) or greater. 0 indicates that there is no limit.	Truncates the execution log when it goes beyond this many lines.
<code>max_log_timeout_minutes</code>	Any value of 0 (zero) or greater. 0 indicates that there is no timeout.	Removes lines from each log entry if it is older than this many minutes.
<code>object_timeout_minutes</code>	Any value of 0 (zero) or greater. 0 indicates that there is no timeout.	Removes entries from the list if they are older than this many minutes.

## slave-server-config.xml

```
<slave_config>
  <max_log_lines>0</max_log_lines>
  <max_log_timeout_minutes>0</max_log_timeout_minutes>
  <object_timeout_minutes>0</object_timeout_minutes>
</slave_config>
```



## Use Checkpoints to Restart Jobs

---

Checkpoints let you restart jobs that fail without you having to rerun the entire job from the beginning. You add checkpoints at hops that connect one job entry to another. Then, when you run the job and a checkpoint is encountered, the software saves the state of the job entry, including the parameters, internal result rows, and result files. If an error occurs that causes the job to fail, like the database is not functioning properly so you can't connect to it, the job ends. But, when the error is resolved and you restart the job, instead of the job starting at the beginning, it starts at the last checkpoint before the error occurred. Because the state of the job entry performed up to the checkpoint was saved, the job resumes as if the failure had not occurred. In addition to setting up checkpoints, you need to set up a checkpoint log. The checkpoint log tracks each time a job runs and where failures occur. This can be helpful for troubleshooting.

In addition to the general instructions presented in this section, [checkpoints are addressed in detail in the wiki](#).

- [Add a Checkpoint](#)
- [Delete a Checkpoint](#)
- [Set Up a Checkpoint Log](#)

### Add a Checkpoint

To add a checkpoint, complete these steps.

1. In Spoon, open a job.
2. Right-click a step or transformation in the job, then select **Mark as Checkpoint** from the menu that appears.
3. The checkpoint is added to the job.

### Delete a Checkpoint

To delete a checkpoint, complete these steps.

1. In Spoon, open a job.
2. Right-click a step or transformation in the job, then select **Clear Checkpoint Marker** from the menu that appears.
3. The checkpoint is cleared from the job.

### Set Up a Checkpoint Log

To set up a checkpoint log, complete these steps. Checkpoints logs are covered in detail [in the wiki](#).

1. In Spoon, open a job.
2. Right-click in an empty space in the job's tab that appears in the Spoon window. In the **Job Properties** window, click the **Log** tab.
3. Select **Checkpoints log table**.
4. Add the log connection and the name of the log table, as well as other information as needed.
5. Choose the log table fields you want enabled.

6. When complete, click **OK**.

## Data Integration Operations Mart

---

The DI Operations Mart enables you to collect and query Data Integration log data and then use BA server tools to examine log data in reports, charts, or dashboards.

### Description

The DI Operations Mart is a centralized data mart that stores job or transformation log data for easy reporting and analysis. The data mart is a collection of tables organized as a data warehouse using a star schema. Together, dimension tables and a fact table represent the logging data. These tables need to be created in the DI Operations Mart database. Pentaho provides SQL scripts to create these tables for MySQL, Oracle, and PostgreSQL databases. A Data Integration job populates the time and date dimensions.

Note: For optimal performance, be sure to clean the operations mart periodically.

### Getting Started

These procedures describe how to install, set up, and configure the DI Operations Mart to collect ETL logging information. These instructions only apply if you do not have the BA Server. If you are a Pentaho Business Analytics customer, these tables and connections are created automatically.

- [Install the DI Operations Mart](#)
- [Set Up Database Connections](#)
- [Create the DI Operations Mart](#)
- [Configure Logging Data Collection](#)
- [Update the Logging for the DI Operations Mart](#)
- [Load the Sample Data in the DI Operations Mart](#)
- [Set Up and Distribute the Data Models](#)
- [Give Users Access to the DI Operations Mart](#)
- [Create Charts, Reports, and Dashboards Using PDI Operations Mart Data](#)
- [Logging Tables Status for the Data Integration Operations Mart](#)
- [Logging Dimensions and Metrics for the Data Integration Operation Mart](#)
- [Clean Up Operations Mart Tables](#)

## Install the DI Operations Mart

---

These instructions are for customers who are only using the DI Server and its tools. If you purchased Pentaho Business Analytics, the Data Integration (DI) Operations Mart installs as part of the installation process and you do not need to follow these instructions.

1. Navigate to `data-integration-server/pentaho-solutions`.
2. Unzip `pentaho-operatiuons-mart.zip` to a temporary location. The archive folder structure is:
  - `pdi-operations-mart/`
    - `DDL/`
    - `etl/`
    - `models/`
    - `samples/`
      - `dashboards/`
      - `datamart/`
      - `jobs_transformations/`
      - `reports/`
3. On the computer that runs the DI repository from PDI/Spoon, create a folder within the repository, such as `public/pdi_operations_mart`. Import the `etl/pdi-operations-mart.xml` to this folder. The import creates two database connections:
  - `live_logging_info`: The online database that stores the logging data for the sample jobs and transformations.
  - `PDI_Operations_Mart`: The off-line database that contains the DI Operations Mart database objects.

## Set Up Database Connections

---

In order to fetch logging data from the online database, you must define a database connection and populate the Operations Mart. [Installing the DI Operations Mart](#) defines two default database connections:

`live_logging_info`: connects to the online database used for the sample jobs and transformations, and `PDI_Operations_Mart` connects to the off-line DI Operations Mart database. The connection information needs to match your databases. This procedure explains how to define your database connections.

1. From within Spoon, close any jobs or transformations you might have open.
2. Select **Tools > Repository > Explore** from the drop-down menu. The **Repository Explorer** window opens.
3. Select the **Connections** tab.
4. Select the `live_logging_info` entry. Edit the entry by clicking the pencil icon in the upper-right corner. The database connection window opens.
5. Enter the information for the connection that fetches the online logging data. If you need to have multiple connections to fetch the online logging information, create new connections and enter the information for each connection.
6. Repeat steps 3 and 4 for the `PDI_Operations_Mart` connection.

The database connections for the DI Operations Mart are defined.

## Create the DI Operations Mart

---

1. From any program that can run scripts against the logging data database, execute the DDL script called `pentaho_mart_<database>.sql`, where `<database>` is the database vendor, such as MySQL, Oracle, PostgreSQL, and alike. These tables are created:
  - `dim_batch`
  - `dim_date`
  - `dim_execution`
  - `dim_executor`
  - `dim_log_table`
  - `dim_step`
  - `dim_time`
  - `fact_execution`
  - `fact_jobentry_execution`
  - `fact_perf_execution`
  - `fact_step_execution`
2. From within Spoon, run the job named `Fill` in `DIM_DATE` and `DIM_TIME`. The DI Operations Mart is created.

## Configure Logging Data Collection

---

For each job or transformation, use this procedure to configure which logging data you would like to collect.

You must set the global logging variables for your database or table in advance. See *Setting Global Logging Variables* for information about how to set global logging variables for transformations and jobs. If you have already set the global logging variables, start at Step 6 to specify an input and output step for each transformation that collects external input or output data.

1. From within Spoon, open a job or transformation.
2. Select the **View** tab, then right-click the job or transformation and select **Settings**.
3. Select the **Logging** tab and choose the appropriate selection from the left pane.
  - For jobs, select **Job**.
  - For transformations, select **Transformation**.
4. In the **Log Connection** field, enter or select the appropriate [database connection](#). Using the provided samples, select the connection named `live_logging_info`.
5. In the **Log table name** field
  - For jobs, enter `LOG_JOB`.
  - For transformations, enter `LOG_TRANS`.
6. In order to collect row input or output information for jobs or transformations, for instance for throughput calculations, specify an input and output step for each transformation that collects external input or output data.
  - For `LINES_INPUT`, specify the step collecting external input data.
  - For `LINES_OUTPUT`, specify the step collecting output data.
7. Ensure all entries under **Log table** fields are selected. If the `LOG_JOB` or `LOG_TRANS` table has not been created for the specified database, click the **SQL** button and then click the **Execute** button in the subsequent dialog box.
8. Click **OK** until you return to the **Job/Transformation properties** dialog box.
9. From the **Monitoring** tab, check the box labeled **Enable step performance monitoring?** Click **OK** to exit the dialog box, then save the job or transformation.

The DI Operations Mart is configured to collect ETL logging data.

## Update the Logging for the DI Operations Mart

You can monitor the latest performance of your ETL operations by updating the logging data within the DI Operations Mart. As a prerequisite, the Operations Mart must have previously been [created](#) and configured with the [logging data](#) you want to collect.

Your data logs need to be updated if you modified these types of data.

- Logging table
- Database connection
- Transformation step
- Job entry

You must update and then populate the executor and log dimensions table if you want to log the most current data. You also have the option to only update or populate.

1. From within Spoon, select **Tools > Repository > Explore**.
2. Select **pdi\_operations\_mart**.
3. Choose the appropriate job or transformation from the table.

If you want to	Choose
Update the executor and log dimension tables.	<b>Update Executor and Log Table Dimensions.ktr</b>
Populate the Pentaho Operations Mart with the logging information <i>without</i> updating executor and log dimension tables.	<b>Update_Logging_Datamart.kjb</b>
Update the executor and log dimension tables with the latest logging data. Then, update the Pentaho Operations Mart with that new data.	<b>Update_Dimensions_then_Logging_Datamart.kjb</b>

As a result, the job or transformation runs. The Operations Mart updates and/or populates with the latest logging data.



## Load the Sample Data in the DI Operations Mart

---

For the sake of illustration, this procedure uses the `Fill_datamart_with_sample_data.kjb` job in the `pdi-operations-mart\samples\datamart` folder to load sample data. This job is intended to be run from the file system; *do not* import it into your EE repository.

**CAUTION:**

If you have loaded your own data to the `PDI_Operations_Mart`, this procedure will corrupt the data.

Transformations use the `PDI_Operations_Mart` database connection. The connection information has been left out. Use this procedure to configure the connection so that all transformations can use it.

1. Open the transformation and edit the `PDI_Operations_Mart`, adding the correct database information.
2. From the transformations **View** pane to the left, right-click the database connection you just edited to bring up its context menu. Then click **Share** and close the transformation. Close the job to ensure the change is saved. Only the `Fill_datamart_with_sample_data.kjb` job should be open now within Spoon.
3. From within Spoon, execute the `Fill_datamart_with_sample_data.kjb` job.

The sample data is loaded into the DI Operations Mart.

- [Loading the Sample Reports, Charts, and Dashboards](#)

## Loading the Sample Reports, Charts, and Dashboards

---

This procedure explains how to load and view sample reports, charts, and dashboards, which were created using the Data Integration Operations Mart under the BA Server.

1. From within the User Console, create a folder under the BA Server at `biserver-ee\pentaho-solutions\`. The internal location under the BA Server is `biserver-ee\pentahosolutions\`. In the Solution pane, name the folder `PDI Operations Mart Sample Reports`.
2. Copy the contents of the `/pdi-operations-mart/samples/reports` directory into the `PDI Operations Mart Sample Reports` folder.
3. Open the Browse view, then select **Tools > Refresh > Repository cache**. Then select **Tools > Refresh > Reporting Metadata**.

Sample reports appear in the **Files** area of the **Browse** view.

Repeat this procedure for loading sample charts or dashboards by substituting either for the word *reports* in step 2.

## Set Up and Distribute the Data Models

---

This procedure describes how to set up and distribute the data models for the DI Operations Mart under the BA Server:

1. Place the `models/[MyBusinessModel].xml` and the `models/PDI_Operations_Mart_<dbname>.mondrian.xml`, where `<dbname>` is the name of the database containing the DI Operations Mart, under the BA Server directory at the following location: `bi-server/pentaho_solutions/PDI Operations Mart Sample Reports`. You may also put the `[MyBusinessModel]` into `bi-server/pentaho_solutions/admin/resources/metadata` or to another solutions directory so it is available independent of the sample solution. In case you already have an existing `[MyBusinessModel]` in this location, you need to rename the file to `PDI_Operations_Mart.xml`. The same applies accordingly to the Mondrian schema.
2. Rename the `PDI_Operations_Mart_<dbname>.mondrian.xml` file to `PDI_Operations_Mart.mondrian.xml`.
3. [Create a new database connection](#). Name it `PDI_Operations_Mart`. Then set the connection information to the Data Integration Operations Mart database location. You can copy the needed URL from the database connecting within Spoon when editing the connection information and pressing the **Feature List** button.
4. Add a category entry to the `datasources.xml` file, which can be found here: `bi-server/pentaho_solutions/system/olap/`.

If you named your data source `PDI_Operations_Mart` and used the solution folder `PDI Operations Mart Sample Reports`, then the addition to the XML file looks like this.

```
<Catalog name="PDI_Operations_Mart">
  <DataSourceInfo>Provider=mondrian;DataSource=PDI_Operations_
Mart</DataSourceInfo>
  <Definition>
    solution:PDI Operations Mart Sample Reports/PDI_Operations_Mart.
mondrian.xml
  </Definition>
</Catalog>
```

5. Save the file.
6. Restart the BA Server.
7. From the User Console, refresh all caches and settings.

The DI Operations Mart data models are available for Analyzer and Interactive reporting using the `PDI_Operations_Mart` data source.

## Give Users Access to the DI Operations Mart

---

You must have previously mapped user roles, as described in [Mondrian Role Mapping in the BA Server](#).

By default, only users who have the **Admin** role can access the Pentaho Operations Mart. The **Admin** role has access to all capabilities within all Pentaho products, including the Pentaho Operations Mart. If you want to allow users to view and run the Pentaho Operations Mart only, you can assign them the **Pentaho Operations** role. For example, a user who has been assigned the **Pentaho Operations** user role is able to open and view a report within the DI Operations mart, but does not have the ability to delete it.

To give users access to view the DI Operations Mart, assign the Pentaho Operations role to those users.

1. From within the Pentaho User Console, select the **Administration** tab.
2. From the left panel, select **Security > Users/Roles**.
3. Select the **Roles** tab.
4. Add the new role called **Pentaho Operations** by following the instructions in [Adding Roles](#).
5. Assign the appropriate users to the new role, as described in [Adding Users to Roles](#).
6. Advise these users to log in to the Pentaho User Console, create a Pentaho Analyzer or Pentaho Interactive Report, and ensure that they can view the Pentaho Operations Mart in the **Select a Data Source** dialog.

## Create Charts, Reports, and Dashboards Using PDI Operations Mart Data

---

Once you have created and populated your Data Integration Operations Mart with log data, the features of the User Console enable you to examine this data and create reports, charts, and dashboards. We provide many samples, reports, charts, and dashboards that you can modify to include your own log data.

### For More Information

For more information about creating or modifying reports, charts, or dashboards, see [Create Analysis, Interactive Reports, and Dashboards](#).

- [Create ETL Logging Reports](#)
- [Create ETL Logging Charts](#)
- [Create ETL Logging Dashboards](#)

## Create ETL Logging Reports

---

1. Open a new report in Report Designer.
2. Create two parameters named `Start Date` and `End Date`, both with a **Value Type** of `Date` and a **Prompt Display Type** of `Date Picker`.
3. Create a parameter named `Number of Rows` with a **Value Type** of `Integer` and a default value of 50.
4. Create a table data set named `PeriodSelection` with an **ID** column of type `Integer` and a **Value** column of type `String`. Enter these ID/Value pairs into the table.
  - 1, "24 Hours"
  - 7, "7 Days"
  - 30, "30 Days"
5. Create a parameter named `Period Selection` with the these settings.

- Value Type = `Integer`
- Prompt Display Type = `Single Selection Button`
- Query = `PeriodSelection`
- Prompt Value = `ID`
- Prompt Display Name = `Value`

Check the **Validate Values** and **Use first value if default value formula results in NA** boxes.

6. Create a new metadata data set. In the Metadata Data Source editor under **XMI file**, point to the metadata file in the solutions folder under the BA Server at `biserver-ee/pentaho-solutions/PDI Operations Mart Sample Reports/[MyBusinessModel.xml]`.
7. Create a query against the metadata data set named `Status` and add the following field to the **Selected Columns** list: **Dim execution > Execution status**.
8. Add a parameter named `Status Selection` with the these settings.

- Value Type = `String`
- Default Value = `[start,end]`
- Prompt Display Type = `Multivalue List`
- Query = `Status`
- Prompt Value = `Execution Status`
- Prompt Display Name = `Execution Status`

Check the **Validate Values** and **Use first value if default value formula results in NA** boxes.

9. Create a query against the metadata data set named `TypeSelection`, add the **Dim executor > Executor type** field to the **Selected Columns** list. Add the following condition: `Dim executor > Executor type is not null`.
10. Add a parameter named with these settings.
  - Value Type = `String`

- Default Value = [job,transformation]
- Prompt Display Type = Multi Selection Button
- Query = TypeSelection
- Prompt Value = Executor type
- Prompt Display Name = Executor type

Check the **Validate Values** and **Use first value if default value formula results in NA** boxes.

11. Create a query against the Metadata data set named `LastRun` and add these fields to the **Selected Columns** list.

- Dim executor > Executor name
- Fact execution > Executor timestamp
- Dim execution > Execution status
- Fact execution > Duration
- Dim executor > Executor type

12. Add these conditions to the query.

- Dim execution > Execution status in {Status Selection}, with default value "start|end"
- Dim executor > Executor type in {Kettle Type}, with default value "transformation"
- Fact execution > Execution Timestamp >= {Start Date}
- Fact execution > Execution Timestamp <= {End Date}

13. Add the following order to the query: Fact execution > Execution timestamp (Descending - DESC).

14. Click **OK** twice to exit the Query Editor and the Metadata Data Source Editor.

15. Drag a **Message** field from the panel on the left onto the report under **Report Header**, enter `Last Run Jobs and Transformations` and format as necessary.

16. Drag 5 **Message** fields onto the **Group Header** band and fill them with the this text.

- Date/Time of Execution
- Name of Job or Transformation
- Type
- Execution Status
- Duration (sec)

Format as necessary.

17. Drag the these fields onto the **Details** band and fill them with the corresponding values.

- **Date** field: Execution Timestamp
- **String** field: Executor Name
- **String** field: Executor Type
- **String** field: Execution Status
- **Number** field: Duration

Align the field widths to the **Group Header** message field widths, in order to align the headers with the values.

18. Review the report, selecting various parameter values to verify the report is working correctly.



## Create ETL Logging Charts

---

1. Open a new report from within Pentaho Report Designer.
2. Create two parameters: `Start Date` and `End Date`, both with a **Value Type** of `Date` and a **Prompt Display Type** of `Date Picker`.
  - a. From within Pentaho Report Designer, Go to **Data > Add Parameter**. This brings up the **Add Parameter** dialog box.
  - b. For the **Name** field, enter `Start Date`.
  - c. Set the **Value Type** to `Date`.
  - d. Set the **Display Type** to `Date Picker`.
  - e. Repeat steps 2a. through 2d, but name this second parameter `End Date`.
3. Create a new metadata data set. In the Metadata Data Source editor, under **XMI file**, point to the metadata file in the solutions folder under the BA Server at `biserver-ee/pentaho-solutions/PDI Operations Mart Sample Reports/metadata.xmi`.
  - a. Go to **Data > Add Datasource > Metadata**. This brings up the Metadata Data Source editor dialog box.
4. Set the **Domain Id / BI-Server Solution Name:** to `PDI Operations Mart Sample Reports/metadata.xmi`
5. Create a new query named `Top 10 Slowest Transformations`, then open the Metadata Query Editor.
6. Add these fields to the **Selected Columns** list.
  - `Dim executor > Executor name`
  - `Fact execution > Minimum Throughput`
7. Add these conditions to the query:.
  - `Dim executor > Executor type = 'transformation'`
  - `Fact execution > Minimum Throughput > 0`
  - `Fact execution > Execution Timestamp >= {Start Date}`
  - `Fact execution > Execution Timestamp <= {End Date}`
8. Add the following order to the query: `Fact execution > Minimum Throughput (Ascending - ASC)`
9. Click **OK** twice to exit the Query Editor and the Metadata Data Source Editor.
10. Create a new Open Formula. Name it `rc` and set the value to: `&{RCOUNT([LC_Dim_executor_executor_name])+1}& " : " & [LC_Dim_executor_executor_name]`.
11. Create a new chart by dragging the chart icon on the left to the **Report Header** section of the report. Set the size.
12. Double-click the chart to bring up the **Edit Chart** dialog box. Select **XY Line Chart** as the chart type.
13. Select **CategorySet Collector** as the collector type under **Primary Datasource**.

14. Set the **category-column** value to the **rc** function, and the **value-columns** value to the **Minimum Throughput** field.
15. Under **Bar Chart properties**, set these options.
  - **x-axis-label-rotation** to 45
  - **show-legend** to `False`
  - **tooltip-formula** to `["chart::category-key"]`
16. Preview the report, selecting various parameter values to verify the chart is being displayed correctly.

## Create ETL Logging Dashboards

---

This procedure shows how to create a dashboard using the sample log data and report created in the [Loading Sample Reports, Charts, and Dashboards](#) procedure.

Dashboards and the elements within them are highly configurable and can be modified to fit a variety of complex analytical needs. See [Use Dashboard Designer](#) for more information about creating and customizing dashboards.

1. From with User Console, click on **New Dashboard** icon.
2. Select the 2 and 1 template.
3. From the **Files** pane on the left, drag these items from the list to the dashboard canvas.
  - Top 10 Failed Jobs and Transformations
  - Top 10 Slowest Transformations
  - Transformation Throughput Overview
4. For each content pane do the following:
  - a. Give the content pane an appropriate title.
  - b. From the **Objects** pane, select **Prompts**.
  - c. Select **Add** to add the first of two new prompts.
  - d. Name the first prompt `Start Date` and make it a **Date Picker**. Click **OK**.
  - e. Name the next prompt `Period` and make it a button. Set the **Data Type** to **Static List**. Click **OK**.
5. Save the dashboard.
6. Select the **Pencil** icon on the toolbar to place the dashboard in edit mode.

You have a dashboard that displays log data created using the Data Integration Operations Mart.

# Logging Tables Status for the Data Integration Operations Mart

---

## Transformation Log Tables

The transformation tables have a **status** column, these are descriptions of the values that can be found in that column.

Status Display	Description
start	Indicates the transformation was started and remains in this status until the transformation ends when no logging interval is set.
end	Transformation ended successfully.
stop	Indicates the user stopped the transformation.
error	Indicates an error occurred when attempting to run the transformation.
running	A transformation is only in this status directly after starting and does not appear without a logging interval.
paused	Indicates the transformation was paused by the user and does not appear without a logging interval.

## Jobs Log Tables

The job log tables have a **status** column, these are descriptions of the values that can be found in that column.

Status Display	Description
start	Indicates the job was started and keeps in this status until the job ends, and when no logging interval is set.
end	Job ended successfully.
stop	Indicates the user stopped the job.
error	Indicates an error occurred when attempting to run the job.
running	A job is only in this status directly after starting and does not appear without a logging interval.
paused	Indicates the job was paused by the user, and does not appear without a logging interval.

## Logging Dimensions and Metrics for the Data Integration Operation Mart

These tables are references that identify the various dimensions and metrics that can be used to create new ETL log charts and reports.

### Fact Table

(fact\_execution)

Field Name	Description
execution_date_tk	A technical key (TK) linking the fact to the date when the transformation/job was executed.
execution_time_tk	A technical key (TK) linking the fact to the time-of-day when the transformation/job was executed.
batch_tk	A technical key (TK) linking the fact to batch information for the transformation/job.
execution_tk	A technical key (TK) linking the fact to execution information about the transformation/job.
executor_tk	A technical key (TK) linking the fact to information about the executor (transformation or job).
parent_executor_tk	A technical key (TK) linking the fact to information about the parent transformation/job.
root_executor_tk	A technical key (TK) linking the fact to information about the root transformation/job.
execution_timestamp	The date and time when the transformation/job was executed.
duration	The length of time (in seconds) between when the transformation was logged (LOGDATE) and the maximum dependency date (DEPDATE)
rows_input	The number of lines read from disk or the network by the specified step. Can be input from files, databases, etc.
rows_output	The number of rows output during the execution of the transformation/job.
rows_read	The number of rows read in from the input stream of the the specified step.
rows_written	The number of rows written during the execution of the transformation/job.
rows_rejected	The number of rows rejected during the execution of the transformation/job.

Field Name	Description
errors	The number of errors that occurred during the execution of the transformation/job.

## Batch Dimension

(dim\_batch)

Field Name	Description
batch_tk	A technical key (TK) for linking facts to batch information.
batch_id	The ID number for the batch.
logchannel_id	A string representing the identifier for the logging channel used by the batch.
parent_logchannel_id	A string representing the identifier for the parent logging channel used by the batch.

## Date Dimension

(dim\_date)

Field Name	Description
date_tk	A technical key (TK) for linking facts to date information.
date_field	A Date object representing a particular day (year, month, day).
ymd	A string representing the date value in year-month-day format.
ym	A string representing the date value in year-month format.
year	An integer representing the year value.
quarter	An integer representing the number of the quarter (1-4) to which this date belongs.
quarter_code	A 2-character string representing the quarter (Q1-Q4) to which this date belongs.
month	An integer representing the number of the month (1-12) to which this date belongs.
month_desc	A string representing the month ("January".."December") to which this date belongs.
month_code	A string representing the shortened month code ("JAN".."DEC") to which this date belongs.

Field Name	Description
day	An integer representing the day of the month (1-31) to which this date belongs.
day_of_year	An integer representing the day of the year (1-366) to which this date belongs.
day_of_week	An integer representing the day of the week (1-7) to which this date belongs.
day_of_week_desc	A string representing the day of the week ("Sunday".. "Saturday") to which this date belongs.
day_of_week_code	A string representing the shortened day-of-week code ("SUN".. "SAT") to which this date belongs.
week	An integer representing the week of the year (1-53) to which this date belongs.

## Execution Dimension

(dim\_execution)

Field Name	Description
execution_tk	A technical key (TK) for linking facts to execution information.
execution_id	A unique string identifier for the execution.
server_name	The name of the server associated with the execution.
server_host	The name of the server associated with the execution.
executing_user	The name of the user who initiated the execution.
execution_status	The status of the execution (start, stop, end, error).

## Executor Dimesion

This table contains information about an executor that is a job or transformation (dim\_executor).

Field Name	Description
executor_tk	A technical key (TK) for linking facts to executor information
version	An integer corresponding to the version of the executor
date_from	A date representing the minimum date for which the executor is valid
date_to	A date representing the maximum date for which the executor is valid
executor_id	A string identifier for the executor

Field Name	Description
executor_source	The source location (either file- or repository-relative) for the executor
* executor_environment	File server, repository name, related to the executor_source. <i>*Reserved for future use.</i>
executor_type	The executor type ("job" or "transformation")
executor_name	The name of the executor (transformation name, e.g.)
executor_desc	A string description of the executor (job description, e.g.)
executor_revision	A string representing the revision of the executor ("1.3", e.g.)
executor_version_label	A string representing a description of the revision (i.e. change comments)
exec_enabled_table_logging	Whether table logging is enabled for this executor. Values are "Y" if enabled, "N" otherwise.
exec_enabled_detailed_logging	Whether detailed (step or job entry) logging is enabled for this executor. Values are "Y" if enabled, "N" otherwise.
exec_enabled_perf_logging	Whether performance logging is enabled for this executor. Values are "Y" if enabled, "N" otherwise.
exec_enabled_history_logging	Whether historical logging is enabled for this executor. Values are "Y" if enabled, "N" otherwise.
last_updated_date	The date the executor was last updated
last_updated_user	The name of the user who last updated the executor

## Log Table Dimension

This is a "junk dimension" containing log table information (dim\_log\_table).

Field Name	Description
log_table_tk	A technical key (TK) for linking.
object_type	The type of PDI object being logged ("job", "transformation", "step", e.g.)
table_connection_name	The name of the database connection corresponding to the location of the transformation/job logging table
table_name	The name of the table containing the transformation/job logging information
schema_name	The name of the database schema corresponding to the location of the transformation/job logging table
step_entry_table_conn_name	The name of the database connection corresponding to the location of the step/entry logging table



Field Name	Description
step_entry_table_name	The name of the table containing the step/entry logging information
step_entry_schema_name	The name of the database schema corresponding to the location of the step/entry logging table
perf_table_conn_name	The name of the database connection corresponding to the location of the performance logging table
perf_table_name	The name of the table containing the performance logging information
perf_schema_name	The name of the database schema corresponding to the location of the performance logging table

## Time-Of-Day-Dimension

This dimension contains entries for every second of a day from midnight to midnight (dim\_time).

Field Name	Description
time_tk	A technical key (TK) for linking facts to time-of-day information
hms	A string representing the time of day as hours-minutes-seconds ("00:01:35", e.g.)
hm	A string representing the time of day as hours-minutes ("23:59", e.g.)
ampm	A string representing whether the time-of-day isAM or PM. Values are "am" or "pm".
hour	The integer number corresponding to the hour of the day (0-23)
hour12	The integer number corresponding to the hour of the day with respect toAM/PM (0-11)
minute	The integer number corresponding to the minute of the hour (0-59)
second	The integer number corresponding to the second of the minute (0-59)

## Step Fact Table

This fact table contains facts about individual step executions (fact\_step\_execution).

Field Name	Description
execution_date_tk	A technical key (TK) linking the fact to the date when the step was executed.

Field Name	Description
execution_time_tk	A technical key (TK) linking the fact to the time-of-day when the step was executed.
batch_tk	A technical key (TK) linking the fact to batch information for the step.
executor_tk	A technical key (TK) linking the fact to information about the executor (transformation).
parent_executor_tk	A technical key (TK) linking the fact to information about the parent transformation.
root_executor_tk	A technical key (TK) linking the fact to information about the root transformation/job.
execution_timestamp	The date and time when the step was executed.
step_tk	A technical key (TK) linking the fact to information about the step.
step_copy	The step copy number. This is zero if there is only one copy of the step, or (0 to N-1) if N copies of the step are executed.
rows_input	The number of lines read from disk or the network by the step. Can be input from files, databases, etc.
rows_output	The number of lines written to disk or the network by the step. Can be output to files, databases, etc.
rows_read	The number of rows read in from the input stream of the step.
rows_written	The number of rows written to the output stream of the step.
rows_rejected	The number of rows rejected during the execution of the step.
errors	The number of errors that occurred during the execution of the step.

## Step Dimension

This dimension contains information about individual steps and job entries (dim\_step) .

Field Name	Description
step_tk	A technical key (TK) for linking facts to step/entry information
step_id	The string name of the step/entry
* original_step_name	The name of the step/entry template used to create this step/entry ("Table Input", e.g.). <i>*Reserved for future use.</i>

## Job Entry Fact Table

This fact table contains facts about individual job entry executions (fact\_jobentry\_execution).

Field Name	Description
execution_date_tk	A technical key (TK) linking the fact to the date when the job entry was executed.
execution_time_tk	A technical key (TK) linking the fact to the time-of-day when the job entry was executed.
batch_tk	A technical key (TK) linking the fact to batch information for the job entry.
executor_tk	A technical key (TK) linking the fact to information about the executor (transformation or job).
parent_executor_tk	A technical key (TK) linking the fact to information about the parent transformation/job.
root_executor_tk	A technical key (TK) linking the fact to information about the root transformation/job.
step_tk	A technical key (TK) linking the fact to information about the job entry.
execution_timestamp	The date and time when the job entry was executed.
rows_input	The number of lines read from disk or the network by the job entry. Can be input from files, databases, etc.
rows_output	The number of lines written to disk or the network by the job entry. Can be output to files, databases, etc.
rows_read	The number of rows read in from the input stream of the job entry.
rows_written	The number of rows written to the output stream of the job entry.
rows_rejected	The number of rows rejected during the execution of the job entry.
errors	The number of errors that occurred during the execution of the job entry.
result	Whether the job entry finished successfully or not. Values are "Y" (successful) or "N" (otherwise).
nr_result_rows	The number of result rows after execution.
nr_result_files	The number of result files after execution.

## Execution Performance Fact Table

This fact table contains facts about the performance of steps in transformation executions (fact\_perf\_execution).

Field Name	Description
execution_date_tk	A technical key (TK) linking the fact to the date when the transformation was executed.
execution_time_tk	A technical key (TK) linking the fact to the time-of-day when the transformation was executed.
batch_tk	A technical key (TK) linking the fact to batch information for the transformation.
executor_tk	A technical key (TK) linking the fact to information about the executor (transformation).
parent_executor_tk	A technical key (TK) linking the fact to information about the parent transformation/job.
root_executor_tk	A technical key (TK) linking the fact to information about the root transformation/job.
step_tk	A technical key (TK) linking the fact to information about the transformation/job.
seq_nr	The sequence number. This is an identifier differentiating performance snapshots for a single execution.
step_copy	The step copy number. This is zero if there is only one copy of the step, or (0 to N-1) if N copies of the step are executed.
execution_timestamp	The date and time when the transformation was executed.
rows_input	The number of rows read from input (file, database, network, ...) during the interval
rows_output	The number of rows written to output (file, database, network, ...) during the interval
rows_read	The number of rows read from previous steps during the interval.
rows_written	The number of rows written to following steps during the interval.
rows_rejected	The number of rows rejected by the steps error handling during the interval.
errors	The number of errors that occurred during the execution of the transformation/job.
input_buffer_rows	The size of the step's input buffer in rows at the time of the snapshot.
output_buffer_rows	The size of the output buffer in rows at the time of the snapshot.

## Clean Up Operations Mart Tables

---

Cleaning up the PDI Operation Mart consists of running a job or transformation that deletes data older than the specified maximum age. The transformation and job for cleaning up the PDI Operations Mart can be found in the "etl" folder.

1. From within PDI/Spoon, open either `Clean_up_PDI_Operations_Mart.kjb` for jobs or the `Clean_up_PDI_Operations_Mart_fact_table.ktr` for transformations.
2. Set these parameters.
  - **max.age** (required)—the maximum age in days of the data. Job and transformation data older than the maximum age will be deleted from the datamart.
  - **schema.prefix** (optional)—for PostgreSQL databases, enter the schema name followed by a period (.), this will be applied to the SQL statements. For other databases, leave the value blank.

Data that was not within the specified date range is now deleted.

To schedule regular clean up of the PDI Operations Mart, see [Create DI Solutions](#).

## DI Operations Mart

---

The DI Operations Mart uses Spoon transformations and jobs to collect and query Data Integration log data. You can then use BA server tools to visualize log data in reports, charts, or dashboards.

- [Set Up DI Operations Mart](#)
- [Maintain and Update DI Operations Mart](#)
- [Reference Tables for DI Operations Mart](#)

## Set Up DI Operations Mart

---

The DI Operations Mart is a centralized data mart that stores job or transformation log data for easy reporting and analysis. The data mart is a collection of tables organized as a data warehouse using a star schema. Together, dimension tables and a fact table represent the logging data. These tables need to be created in the DI Operations Mart database. Pentaho provides SQL scripts to create these tables for MySQL, Oracle, and PostgreSQL databases. A Data Integration job populates the time and date dimensions.

### Unzip DI Operations Mart Files

The files needed to install the DI Operations Mart are already on your computer. Here is how to find and unzip those files, then set up the database.

1. Unzip the `pentaho-operations-mart-<version number>.zip` file, which is located in the `data-integration-server/pentaho-solutions` directory. `<version number>` indicates the version number Pentaho assigns to the build of the DI Operations Mart. We recommend that you unzip the file in a temporary directory outside of the pentaho directory structure.

2. When unzipped, the directory structure looks like this.

- `pdi-operations-mart/`
- `DDL/`
- `etl/`
- `models/`
- `pentaho_operations_mart/`
- `samples/`
- `exportManifest.xml`

### Create Logging Tables and Datamart

Logging tables must be created in the database.

1. From any program that can run scripts against the logging data database, execute `pentaho_logging_<database>.sql`. (`<database>` is MySQL, Oracle, or PostgreSQL). `pentaho_logging_<database>.sql` is in the `pdi-operations-mart/DDL/<database>` directory. The script generates several logging schemas.
2. If you installed PDI using the custom or manual method, the `pdi-operations-mart` database must be created. Execute the DDL script called `pentaho_mart_<database>.sql`, where `<database>` is the database vendor, such as MySQL, Oracle, or PostgreSQL. The DDL script can be found in the `pdi-operations-mart/DDL/<database>` directory.
3. When complete, the following tables are created.

Dimension Tables	Fact, Pro-Audit, and Staging Tables
<ul style="list-style-type: none"> <li>• dim_batch</li> <li>• dim_component</li> <li>• dim_content_item</li> <li>• dim_date</li> <li>• dim_execution</li> <li>• dim_executor</li> <li>• dim_instance</li> <li>• dim_log_table</li> <li>• dim_session</li> <li>• dim_state</li> <li>• dim_step</li> <li>• dim_time</li> </ul>	<ul style="list-style-type: none"> <li>• fact_component</li> <li>• fact_execution</li> <li>• fact_instance</li> <li>• fact_jobentry_execution</li> <li>• fact_perf_execution</li> <li>• fact_session</li> <li>• fact_step_execution</li> <li>• pro_audit_staging</li> <li>• pro_audit_tracker</li> <li>• stg_content_item</li> </ul>

1. Execute the following SQL script in the tool of your choice. The script enables DI Operations Mart to run on PostgreSQL.

```
ALTER TABLE pentaho_dilogs.jobentry_logs DROP COLUMN result;
ALTER TABLE pentaho_dilogs.jobentry_logs ADD COLUMN "result" boolean;
ALTER TABLE pentaho_operations_mart.fact_jobentry_execution DROP COLUMN result;
ALTER TABLE pentaho_operations_mart.fact_jobentry_execution ADD COLUMN "result"
CHAR(5);
```

## Set Global Kettle Logging Variables

Set up Kettle logging variables so that by default, transformation, job, step, and performance logging data is captured in tables you specify.

1. In Spoon, select **Edit > Edit the kettle.properties** file.
2. In the **Kettle properties** window, set the following variable names and values. The variable names and values in this section are the default. If you have set up your operations mart differently, change the variable names and values so they match your environment.

Variable Name	Value
KETTLE_JOB_LOG_DB	live_logging_info
KETTLE_JOB_LOG_TABLE	job_logs
KETTLE_TRANS_LOG_DB	live_logging_info
KETTLE_TRANS_LOG_TABLE	trans_logs



Variable Name	Value
KETTLE_STEP_LOG_DB	live_logging_info
KETTLE_STEP_LOG_TABLE	step_logs
KETTLE_JOBENTRY_LOG_DB	live_logging_info
KETTLE_JOBENTRY_LOG_TABLE	jobentry_logs
KETTLE_TRANS_PERFORMANCE_LOG_DB	live_logging_info
KETTLE_TRANS_PERFORMANCE_LOG_TABLE	transperf_logs
KETTLE_CHANNEL_LOG_DB	live_logging_info
KETTLE_CHANNEL_LOG_TABLE	channel_logs
KETTLE_METRICS_LOG_DB	live_logging_info
KETTLE_METRICS_LOG_TABLE	metrics_logs

0. 1. If you choose to use a non-default schema, set these variables as well. Their values should match the name of your schema.

- KETTLE\_JOB\_LOG\_SCHEMA
- KETTLE\_TRANS\_LOG\_SCHEMA
- KETTLE\_STEP\_LOG\_SCHEMA
- KETTLE\_JOBENTRY\_LOG\_SCHEMA
- KETTLE\_TRANS\_PERFORMANCE\_LOG\_SCHEMA
- KETTLE\_CHANNEL\_LOG\_SCHEMA
- KETTLE\_METRICS\_LOG\_SCHEMA

7. 1. Click **OK**.

8. 2. Restart Spoon and the DI Server.

## Import DI Operations Mart

Follow these instructions to import DI Operations Mart tables, schema, and database connection into your database.

1. Start Spoon, then connect to the DI Repository.
2. Select **Tools > Repository > Import Repository**.
3. Select `etl/pdi-operations-mart.xml` then click **Open**. The `etl` directory is where you unzipped the `pentaho-operations-mart-<version number>.zip` file.
4. When prompted about setting rules for the import, click **No**.
5. The **Directory Selection** window appears. To create a new directory where the PDI Operations Mart will be stored:

- e. **A.** Right-click the **public** folder and select **New sub-directory**.
  - f. **B.** In the **Enter the Directory Name** window, type **pdi\_operations\_mart** then click **OK**.
  - g. **C.** **pdi\_operations\_mart** appears. Make sure it is highlighted, then click **OK**.
  - h. **D.** Click **OK** in the **Enter comment** window.
6. The **Repository Import** window appears with a status of the import. When the *Import finished* message appears, click the **Close** button.

## Set Up Database Connections

The connection information must match the credentials needed for your database.

1. In Spoon, close open jobs and transformations.
2. Select **Tools > Repository > Explore** from the drop-down menu.
3. In the **Repository Explorer** window, select the **Connections** tab.
4. Create a database connection and name it **live\_logging\_info**. In the **Database Connection** window enter information for the connection that gets the online logging data. The default connection information appears in the following table, but yours might be different, depending on the ports and password assigned during PDI installation.

Item	Default Value
Connection Name	live_logging_info
Hostname	localhost
Database Name	di_hibernate
Port Number	5432
User Name	hibuser
Advanced/Preferred Schema	pentaho_dilogs
Password	See your system administrator for the default value.

1. Click **OK** to save the information.
2. If you need to have multiple connections to get the online logging information, create new connections and enter the information for each connection.
3. Select the **pentaho\_operations\_mart** from the **Connections** tab and edit the entry by clicking the pencil icon in the upper-right corner. The **Database Connection** window opens.
4. Enter the information for the connection. When finished, click **OK**.

Item	Default Value
Connection Name	pentaho_operations_mart

Item	Default Value
Hostname	localhost
Database Name	di_hibernate
Port Number	5432
User Name	hibuser
Advanced/Preferred Schema	pentaho_operations_mart
Password	See your system administrator for the default value.

## Update Links

Update the Fill in DIM\_Date and DIM\_Time job so it points to the version in the repository, then run the Fill in DIM\_Date and DIM\_Time job to finish setting up the DI Operations Mart.

1. In Spoon, select **Tools > Repository > Explore**.
2. In the **Browse** tab, click the **pdi\_operations\_mart**.
3. Double-click Fill in DIM\_Date and DIM\_Time to open it in Spoon.
4. Click **Close** to close the Repository Explorer.
5. Double-click Generate DIM\_DATE.
  - d. **A.** In the **Transformation specification** tab, click **Specify by name and directory**.
  - e. **B.** There are two fields that become active. The first is where you specify the directory. The second is where you specify the name of the transformation. Position the cursor in the first field, then press [CTRL]+[SPACE] and select **\${Internal.Job.Repository.Directory}** from the options that appear.
  - f. **C.** In the second field enter `Generate_DIM_DATE.ktr`.
  - g. **D.** Click **OK**.
6. Double-click Generate DIM\_TIME and repeat step 5a - 5d. For step 5c, enter `Generate_DIM_TIME.ktr` instead of `Generate_DIM_DATE.ktr`.
7. Save the job, then run it.
8. When complete, close the job.

## Test DI Operations Mart

To run a test, build and run a simple transformation or job, then run the DI Operations Mart.

1. In Spoon, create and save a transformation and a job. For more information on how to create a transformation or job, see the Get Started with DI tutorial.
2. Select **Tools > Repository > Explore**, then select **pdi\_operations\_mart**.
3. Select the Update Dimensions then Logging Datamart.kjb job, then run it.

## Schedule DI Operations Mart Job

As a best practice, you should schedule the updating of Pentaho Operations Mart data so that you have up-to-date information for Operations Mart reports. It is recommended that you schedule log updates at least daily to ensure you have the freshest data.

1. In Spoon, select **Tools > Repository > Explore**, then select **pdi\_operations\_mart**.
2. Select the `Update_Logging_Datamart.kjb` job and open it.
3. Set a schedule for this job by following the instructions in Scheduling Transformations and Jobs from Spoon.
4. When finished, close and save the schedule and the job.

## Give Users Access to the DI Operations Mart

You must have previously mapped user roles, as described in Mondrian Role Mapping in the BA Server.

By default, only users who have the **Admin** role can access the Pentaho Operations Mart. The **Admin** role has access to all capabilities within all Pentaho products, including the Pentaho Operations Mart. If you want to allow users to view and run the Pentaho Operations Mart only, you can assign them the **Pentaho Operations** role. For example, a user who has been assigned the **Pentaho Operations** user role is able to open and view a report within the DI Operations mart, but does not have the ability to delete it.

To give users access to view the DI Operations Mart, assign the Pentaho Operations role to those users.

1. From within the **Pentaho User Console**, select the **Administration** tab.
2. From the left panel, select **Security > Users/Roles**.
3. Select the **Roles** tab.
4. Add the new role called **Pentaho Operations** by following the instructions in Adding Roles.
5. Assign the appropriate users to the new role, as described in Adding Users to Roles.
6. Advise these users to log in to the Pentaho User Console, create a Pentaho Analyzer or Pentaho Interactive Report, and ensure that they can view the Pentaho Operations Mart in the **Select a Data Source** dialog.

# Maintain and Update DI Operations Mart

---

## Override Kettle Logging Variable Settings

If you want to override the Kettle logging variable settings for where logging data is stored in a specific transformation or job, complete these steps. Even if you override those settings, logging information will be used to create Operations Mart reports.

1. From within Spoon, open a job or transformation.
2. Select the **View** tab, then right-click the job or transformation and select **Transformation Settings** or **Job Settings**.
3. Select the **Logging** tab and choose the appropriate selection from the left pane.
  - For jobs, select **Job**.
  - For transformations, select **Transformation**.
1. In the **Log Connection** field, enter or select the appropriate database connection. If you are using the default settings, choose `live_logging_info`. Otherwise, enter or choose the database connection that reflects your environment's configuration.
2. In the **Log table name** field
  - For jobs, enter `LOG_JOB`.
  - For transformations, enter `LOG_TRANS`.
1. In order to collect row input or output information for jobs or transformations, for instance for throughput calculations, specify an Input and output step for each transformation that collects external input or output data.
  - For `LINES_INPUT`, specify the step collecting external input data.
  - For `LINES_OUTPUT`, specify the step collecting output data.
1. Ensure all entries under **Fields To Log** are selected. If the `LOG_JOB` or `LOG_TRANS` table has not been created in the database, click the **SQL** button and then click the **Execute** button in the subsequent dialog box.
2. Click **OK**.
3. In the **Monitoring** tab in the **Transformation Properties** or **Job Properties** window, check the box labeled **Enable step performance monitoring?**
4. Click **OK** to exit the dialog box, then save the job or transformation.

The DI Operations Mart is configured to collect ETL logging data.

## Update Operations Mart

You can monitor the latest performance of your ETL operations by updating the logging data within the DI Operations Mart. As a prerequisite, the Operations Mart must have previously been created and configured with the logging data you want to collect.

Your data logs need to be updated if you modified these types of data.

- Logging table
- Database connection
- Transformation step
- Job entry

You must update and then populate the executor and log dimensions table if you want to log the most current data.

If you modified the logging table, database connection, operations mart transformation steps or job entries, you need to manually update the DI operations mart executor, log dimension tables, and then refresh the data.

1. From within Spoon, select **Tools > Repository > Explore**.
2. Select **pdi\_operations\_mart**.
3. Choose the appropriate job or transformation from the table.

If you want to	Choose
Update the executor and log dimension tables.	Update Executor and Log Table Dimensions.ktr
Populate the Pentaho Operations Mart with the logging information <i>without</i> updating executor and log dimension tables.	Update_Logging_Datamart.kjb
Update the executor and log dimension tables with the latest logging data. Then, update the Pentaho Operations Mart with that new data.	Update_Dimensions_then_Logging_Datamart.kjb

1. Schedule the `Update_Logging_Datamart.kjb` job to run periodically. For more information on how to schedule the job, see *Scheduling Transformations and Jobs From Spoon*.

The job or transformation runs. The Operations Mart updates and/or populates with the latest logging data.

## Clean Up Operations Mart Tables

Cleaning up the PDI Operation Mart consists of running a job or transformation that deletes data older than the specified maximum age. The transformation and job for cleaning up the PDI Operations Mart can be found in the "etl" folder.

1. In Spoon, open `Clean_up_PDI_Operations_Mart.kjb`, then set these parameters.

- **max.age** (required)—the maximum age in days of the data. Job and transformation data older than the maximum age will be deleted from the datamart.
  - **schema.prefix** (optional)—for PostgreSQL databases, enter the schema name followed by a period (.), this will be applied to the SQL statements. For other databases, leave the value blank.
1. Data that was not within the specified date range is now deleted.
  2. To schedule regular clean up of the PDI Operations Mart, see Create DI Solutions.

## Reference Tables for DI Operations Mart

---

DI Operations Mart consists of several tables, which are referenced in this section.

### Logging Tables Status for the Data Integration Operations Mart

#### Transformation Log Tables

The transformation tables have a **status** column, these are descriptions of the values that can be found in that column.

Status Display	Description
start	Indicates the transformation was started and remains in this status until the transformation ends when no logging interval is set.
end	Transformation ended successfully.
stop	Indicates the user stopped the transformation.
error	Indicates an error occurred when attempting to run the transformation.
running	A transformation is only in this status directly after starting and does not appear without a logging interval.
paused	Indicates the transformation was paused by the user and does not appear without a logging interval.

#### Jobs Log Tables

The job log tables have a **status** column, these are descriptions of the values that can be found in that column.

Status Display	Description
start	Indicates the job was started and keeps in this status until the job ends, and when no logging interval is set.



Status Display	Description
end	Job ended successfully.
stop	Indicates the user stopped the job.
error	Indicates an error occurred when attempting to run the job.
running	A job is only in this status directly after starting and does not appear without a logging interval.
paused	Indicates the job was paused by the user, and does not appear without a logging interval.

## Logging Dimensions and Metrics for the Data Integration Operation Mart

These tables are references that identify the various dimensions and metrics that can be used to create new ETL log charts and reports.

### Fact Table

(fact\_execution)

Field Name	Description
execution_date_tk	A technical key (TK) linking the fact to the date when the transformation/job was executed.
execution_time_tk	A technical key (TK) linking the fact to the time-of-day when the transformation/job was executed.
batch_tk	A technical key (TK) linking the fact to batch information for the transformation/job.
execution_tk	A technical key (TK) linking the fact to execution information about the transformation/job.
executor_tk	A technical key (TK) linking the fact to information about the executor (transformation or job).
parent_executor_tk	A technical key (TK) linking the fact to information about the parent transformation/job).
root_executor_tk	A technical key (TK) linking the fact to information about the root transformation/job.
execution_timestamp	The date and time when the transformation/job was executed.

Field Name	Description
duration	The length of time (in seconds) between when the transformation was logged (LOGDATE) and the maximum dependency date (DEPDATE)
rows_input	The number of lines read from disk or the network by the specified step. Can be input from files, databases, etc.
rows_output	The number of rows output during the execution of the transformation/job.
rows_read	The number of rows read in from the input stream of the the specified step.
rows_written	The number of rows written during the execution of the transformation/job.
rows_rejected	The number of rows rejected during the execution of the transformation/job.
errors	The number of errors that occurred during the execution of the transformation/job.
failed	Indicates if the job or transformation has failed. 0 means the job or transformation completed successfully. 1 indicates the job or transformation failed.

## Batch Dimension

(dim\_batch)

Field Name	Description
batch_tk	A technical key (TK) for linking facts to batch information.
batch_id	The ID number for the batch.
logchannel_id	A string representing the identifier for the logging channel used by the batch.
parent_logchannel_id	A string representing the identifier for the parent logging channel used by the batch.

## Date Dimension

(dim\_date)

Field Name	Description
date_tk	A technical key (TK) for linking facts to date information.
date_field	A Date object representing a particular day (year, month, day).
ymd	A string representing the date value in year-month-day format.
ym	A string representing the date value in year-month format.
year	An integer representing the year value.
quarter	An integer representing the number of the quarter (1-4) to which this date belongs.
quarter_code	A 2-character string representing the quarter (Q1-Q4) to which this date belongs.
month	An integer representing the number of the month (1-12) to which this date belongs.
month_desc	A string representing the month ("January".."December") to which this date belongs.
month_code	A string representing the shortened month code ("JAN".."DEC") to which this date belongs.
day	An integer representing the day of the month (1-31) to which this date belongs.
day_of_year	An integer representing the day of the year (1-366) to which this date belongs.
day_of_week	An integer representing the day of the week (1-7) to which this date belongs.
day_of_week_desc	A string representing the day of the week ("Sunday".."Saturday") to which this date belongs.
day_of_week_code	A string representing the shortened day-of-week code ("SUN".."SAT") to which this date belongs.
week	An integer representing the week of the year (1-53) to which this date belongs.

## Execution Dimension

(dim\_execution)

Field Name	Description
execution_tk	A technical key (TK) for linking facts to execution information.
execution_id	A unique string identifier for the execution.
server_name	The name of the server associated with the execution.
server_host	The name of the server associated with the execution.
executing_user	The name of the user who initiated the execution.
execution_status	The status of the execution (start, stop, end, error).
client	The name of the client that triggered the execution.

**Executor Dimension** This table contains information about an executor that is a job or transformation (dim\_executor).

Field Name	Description
executor_tk	A technical key (TK) for linking facts to executor information
version	An integer corresponding to the version of the executor
date_from	A date representing the minimum date for which the executor is valid
date_to	A date representing the maximum date for which the executor is valid
executor_id	A string identifier for the executor
executor_source	The source location (either file- or repository-relative) for the executor
* executor_environment	File server, repository name, related to the executor_source. <i>*Reserved for future use.</i>
executor_type	The executor type ("job" or "transformation")
executor_name	The name of the executor (transformation name, e.g.)
executor_desc	A string description of the executor (job description, e.g.)

Field Name	Description
executor_revision	A string representing the revision of the executor ("1.3", e.g.)
executor_version_label	A string representing a description of the revision (i.e. change comments)
exec_enabled_table_logging	Whether table logging is enabled for this executor. Values are "Y" if enabled, "N" otherwise.
exec_enabled_detailed_logging	Whether detailed (step or job entry) logging is enabled for this executor. Values are "Y" if enabled, "N" otherwise.
exec_enabled_perf_logging	Whether performance logging is enabled for this executor. Values are "Y" if enabled, "N" otherwise.
exec_enabled_history_logging	Whether historical logging is enabled for this executor. Values are "Y" if enabled, "N" otherwise.
last_updated_date	The date the executor was last updated
last_updated_user	The name of the user who last updated the executor

## Log Table Dimension

This is a "junk dimension" containing log table information (dim\_log\_table).

Field Name	Description
log_table_tk	A technical key (TK) for linking.
object_type	The type of PDI object being logged ("job", "transformation", "step", e.g.)
table_connection_name	The name of the database connection corresponding to the location of the transformation/job logging table
table_name	The name of the table containing the transformation/job logging information
schema_name	The name of the database schema corresponding to the location of the transformation/job logging table
step_entry_table_conn_name	The name of the database connection corresponding to the location of the step/entry logging table

Field Name	Description
<code>step_entry_table_name</code>	The name of the table containing the step/entry logging information
<code>step_entry_schema_name</code>	The name of the database schema corresponding to the location of the step/entry logging table
<code>perf_table_conn_name</code>	The name of the database connection corresponding to the location of the performance logging table
<code>perf_table_name</code>	The name of the table containing the performance logging information
<code>perf_schema_name</code>	The name of the database schema corresponding to the location of the performance logging table

## Time-Of-Day-Dimension

This dimension contains entries for every second of a day from midnight to midnight (`dim_time`).

Field Name	Description
<code>time_tk</code>	A technical key (TK) for linking facts to time-of-day information
<code>hms</code>	A string representing the time of day as hours-minutes-seconds ("00:01:35", e.g.)
<code>hm</code>	A string representing the time of day as hours-minutes ("23:59", e.g.)
<code>ampm</code>	A string representing whether the time-of-day is AM or PM. Values are "am" or "pm".
<code>hour</code>	The integer number corresponding to the hour of the day (0-23)
<code>hour12</code>	The integer number corresponding to the hour of the day with respect to AM/PM (0-11)
<code>minute</code>	The integer number corresponding to the minute of the hour (0-59)
<code>second</code>	The integer number corresponding to the second of the minute (0-59)

## Step Fact Table

This fact table contains facts about individual step executions (fact\_step\_execution).

Field Name	Description
execution_date_tk	A technical key (TK) linking the fact to the date when the step was executed.
execution_time_tk	A technical key (TK) linking the fact to the time-of-day when the step was executed.
batch_tk	A technical key (TK) linking the fact to batch information for the step.
executor_tk	A technical key (TK) linking the fact to information about the executor (transformation).
parent_executor_tk	A technical key (TK) linking the fact to information about the parent transformation.
root_executor_tk	A technical key (TK) linking the fact to information about the root transformation/job.
execution_timestamp	The date and time when the step was executed.
step_tk	A technical key (TK) linking the fact to information about the step.
step_copy	The step copy number. This is zero if there is only one copy of the step, or (0 to N-1) if N copies of the step are executed.
rows_input	The number of lines read from disk or the network by the step. Can be input from files, databases, etc.
rows_output	The number of lines written to disk or the network by the step. Can be output to files, databases, etc.
rows_read	The number of rows read in from the input stream of the step.
rows_written	The number of rows written to the output stream of the step.
rows_rejected	The number of rows rejected during the execution of the step.
errors	The number of errors that occurred during the execution of the step.

## Step Dimension

This dimension contains information about individual steps and job entries (dim\_step) .

Field Name	Description
step_tk	A technical key (TK) for linking facts to step/entry information
step_id	The string name of the step/entry
* original_step_name	The name of the step/entry template used to create this step/entry ("Table Input", e.g.). <i>*Reserved for future use.</i>

## Job Entry Fact Table

This fact table contains facts about individual job entry executions (fact\_jobentry\_execution).

Field Name	Description
execution_date_tk	A technical key (TK) linking the fact to the date when the job entry was executed.
execution_time_tk	A technical key (TK) linking the fact to the time-of-day when the job entry was executed.
batch_tk	A technical key (TK) linking the fact to batch information for the job entry.
executor_tk	A technical key (TK) linking the fact to information about the executor (transformation or job).
parent_executor_tk	A technical key (TK) linking the fact to information about the parent transformation/job.
root_executor_tk	A technical key (TK) linking the fact to information about the root transformation/job.
step_tk	A technical key (TK) linking the fact to information about the job entry.
execution_timestamp	The date and time when the job entry was executed.
rows_input	The number of lines read from disk or the network by the job entry. Can be input from files, databases, etc.
rows_output	The number of lines written to disk or the network by the job entry. Can be output to files, databases, etc.



Field Name	Description
rows_read	The number of rows read in from the input stream of the job entry.
rows_written	The number of rows written to the output stream of the job entry.
rows_rejected	The number of rows rejected during the execution of the job entry.
errors	The number of errors that occurred during the execution of the job entry.
result	Whether the job entry finished successfully or not. Values are "Y" (successful) or "N" (otherwise).
nr_result_rows	The number of result rows after execution.
nr_result_files	The number of result files after execution.

## Execution Performance Fact Table

This fact table contains facts about the performance of steps in transformation executions (fact\_perf\_execution).

Field Name	Description
execution_date_tk	A technical key (TK) linking the fact to the date when the transformation was executed.
execution_time_tk	A technical key (TK) linking the fact to the time-of-day when the transformation was executed.
batch_tk	A technical key (TK) linking the fact to batch information for the transformation.
executor_tk	A technical key (TK) linking the fact to information about the executor (transformation).
parent_executor_tk	A technical key (TK) linking the fact to information about the parent transformation/job.
root_executor_tk	A technical key (TK) linking the fact to information about the root transformation/job.
step_tk	A technical key (TK) linking the fact to information about the transformation/job.

Field Name	Description
seq_nr	The sequence number. This is an identifier differentiating performance snapshots for a single execution.
step_copy	The step copy number. This is zero if there is only one copy of the step, or (0 to N-1) if N copies of the step are executed.
execution_timestamp	The date and time when the transformation was executed.
rows_input	The number of rows read from input (file, database, network, ...) during the interval
rows_output	The number of rows written to output (file, database, network, ...) during the interval
rows_read	The number of rows read from previous steps during the interval.
rows_written	The number of rows written to following steps during the interval.
rows_rejected	The number of rows rejected by the steps error handling during the interval.
errors	The number of errors that occurred during the execution of the transformation/job.
input_buffer_rows	The size of the step's input buffer in rows at the time of the snapshot.
output_buffer_rows	The size of the output buffer in rows at the time of the snapshot.

## Apply Best Practices for Kettle Logging

---

Kettle logging provides extensive flexibility that allows you to determine log locations, granularity, as well as what information is captured. Here are a best practices for setting up logging in your environment. For more information on logging, see the [Logging and Monitoring Operations](#) article.

- **Store logs in a centralized database.** By default, log files are stored locally. Spoon, Carte, and DI Server logs are stored separately. To make log information easier to find, place logs in a central database. As an added bonus, centralized logging makes it easier to use PDI's performance monitoring more effectively.
- **Use JNDI to Store Database Connection Information.** Although you can create JDBC connections, using JNDI makes it easier for you to switch from development to production environments. Connection details are contained in one external file. For more information on, [see Define JNDI Connections for the DI Server](#).
- **Obtain Full Insert Accesses for Tables.** Logging can fail if you do not have the appropriate accesses. Having the appropriate accesses minimizes this risk. To learn more about table access, consult the documentation for your database.
- **Install JDBC Drivers Locally and on Each Server.** This helps you avoid *Driver not found* errors. [A list of supported drivers appears here](#).
- **Use Implied Schemas When Possible.** Implied schemas result in fewer places to troubleshoot should logging fail. Of course, you can still specify a schema if needed.
- **Make templates for transformation and job files.** Include logging configurations in the template so that they can be reused with ease.
- **Use Kettle Global Logging Variables When Possible.** To avoid the work of adding logging variables to each transformation or job, consider using global logging variables instead. You can override logging variables by adding information to individual transformations or jobs as needed.

If you choose to use the `kettle.properties` file observe the following best practices.

- **Backup your kettle.properties files.** If you are making many changes to the Kettle.properties file, consider backing up the file first. This will make it easier to restore should issues occur.
- **Maintain a Master Copy of the kettle.properties file.** It is usually easiest to use the kettle.properties file for Spoon, then to overwrite the Carte and DI Server copies if changes are made. Make sure that values that might change, such as directory paths, are maintained however.
- **Test Thoroughly.** Test your settings by saving your `kettle.properties` file locally, then restarting spoon. Make sure the `kettle.properties` file loads properly. Execute a transformation or job that uses the settings. Try executing the transformation locally and remotely on the DI Server to ensure that log variables reference the same places.

## Contents of the .kettle Directory

---

File	Purpose
kettle.properties	Main PDI properties file; contains global variables for low-level PDI settings
shared.xml	Shared objects file
db.cache	The database cache for metadata
repositories.xml	Connection details for PDI database or solution repositories
.spoonrc	User interface settings, including the last opened transformation/job
.languageChoice	Default language for the PDI client tool

- [Change the PDI Home Directory Location \(.kettle folder\)](#)

## Change the PDI Home Directory Location (.kettle folder)

---

The default location for the Pentaho Data Integration home directory is the **.kettle** directory in your system user's home directory.

- **Windows:** C:\Documents and Settings\example\_user\.kettle
- **Linux:** ~/.kettle)

There will be a different **.kettle** directory, and therefore a different set of configuration files, for each system user that runs PDI.

### Standalone PDI client tool deployments

You can specify a single, universal **.kettle** directory for all users by declaring a **KETTLE\_HOME** environment variable in your operating system. When declaring the variable, leave out the **.kettle** portion of it; this is automatically added by PDI.

```
export KETTLE_HOME=/home/pentaho/examplepath/pdi
```

### BA Server deployments that run PDI content

If you followed a manual deployment or archive package installation path, you can set a system environment variable as explained above, but it must be declared before the BA Server service starts. You can alternatively change the **CATALINA\_OPTS** system variable to include the **-D** flag for **KETTLE\_HOME**, or you can edit the script that runs the BA Server and set the flag inline, as in this example from the **start-pentaho.sh** script:

```
export CATALINA_OPTS="--Xms2048m -Xmx2048m -XX:MaxPermSize=256m -Dsun.rmi.dgc.client.gcInterval=3600000 -Dsun.rmi.dgc.server.gcInterval=3600000" -DKETTLE_HOME=/home/pentaho/examplepath/pdi
```

### Windows service modification

If you used the graphical utility to install the DI Server, then you must modify the Java options flag that runs the BA Server Tomcat service. Here is an example command that will change the value of **KETTLE\_HOME** to **C:\<examplepath>\pdi\.kettle**:

```
tomcat6.exe //US//pentahobiserver ++JvmOptions -DKETTLE_HOME=C:\examplepath\pdi
```

Modify the **DI Server** in the same way, changing the service name:

```
tomcat6.exe //US//pentahoDataIntegrationServer ++JvmOptions -DKETTLE_  
HOME=C:\<examplepath>\pdi
```

## Troubleshooting

---

This section contains known problems and solutions relating to the procedures earlier.

- [Jobs Scheduled on DI Server Cannot Execute Transformation on Remote Carte Server](#)
- [Sqoop Import into Hive Fails](#)

## Jobs Scheduled on DI Server Cannot Execute Transformation on Remote Carte Server

---

You may see an error like this one when trying to schedule a job to run on a remote Carte server:

```
ERROR 11-05 09:33:06,031 - !UserRoleListDelegate.ERROR_0001_UNABLE_TO_INITIALIZE_
USER_ROLE_LIST_WEBSVC!
com.sun.xml.ws.client.ClientTransportException: The server sent
HTTP status code 401: Unauthorized
```

To fix this, follow the instructions in [Executing Scheduled Jobs on a Remote Carte Server](#).



## Sqoop Import into Hive Fails

---

If executing a Sqoop import into Hive fails to execute on a remote installation, the local Hive installation configuration does not match the Hadoop cluster connection information used to perform the Sqoop job.

Verify the Hadoop connection information used by the local Hive installation is configured the same as the Sqoop job entry.

## Purge Transformations, Jobs, and Shared Content from the DI Repository

The Purge Utility allows you to permanently delete shared objects (servers, clusters, and databases) stored in the DI Repository as well as content (transformations and jobs). You can also delete revision information for content and shared objects.

### CAUTION:

Purging is permanent. Purged items cannot be restored.

To use the Purge Utility, complete these steps.

1. Make sure the DI Repository is running.
2. Open a shell tool, command prompt window, or terminal window, and navigate to the `pentaho/design-tools/data-integration` directory.
3. At the prompt enter the purge utility command. The format for the command, a table that describes each parameter, and parameter examples follow.

### NOTE:

The command must contain the `url`, `user`, and `password` parameters, as well as one of these parameters: `versionCount`, `purgeBeforeDate`, `purgeFiles`, `purgeRevisions`.

### Windows:

```
purge-utility.bat [-url] [-user] [-password] [-purgeSharedObjects] [-versionCount]
[-purgeBeforeDate] [-purgeFiles] [-purgeRevisions] [-logFileName] [-logLevel]
```

### Linux:

```
purge-utility.sh [-url] [-user] [-password] [-purgeSharedObjects] [-versionCount]
[-purgeBeforeDate] [-purgeFiles] [-purgeRevisions] [-logFileName] [-logLevel]
```

Option	Required?	Description
<code>-url</code>	Y	URL address for the DI Repository. This is a required parameter. By default, this the DI Server is installed at this URL: <a href="http://localhost:9080">http://localhost:9080</a> .
<code>-user</code>	Y	Username for an account that can access the DI Server as an administrator. This is a required parameter.
<code>-password</code>	Y	Password for the account used to access the DI Server. This is a required parameter.

-purgeSharedObjects	N	When set to <code>TRUE</code> , the parameter purges shared objects from the repository. This parameter must be used with the <code>purgefile</code> parameter. If you try to purge shared objects without including the <code>purgefile</code> parameter in the command line, an error occurs. If you set the <code>purgeSharedObjects</code> parameter to <code>FALSE</code> , it does not purge shared objects. If you include the <code>purgeSharedObjects</code> parameter in the command, but you don't set it to <code>TRUE</code> or <code>FALSE</code> , the Purge Utility will assume that it is set to <code>TRUE</code> .
-versionCount	<p>You must include <b>only one</b> of these:  <code>versionCount</code>,  <code>purgeBeforeDate</code>,  <code>purgeFiles</code>, or  <code>purgeRevisions</code></p>	Deletes entire version history except the for last <code>versionCount</code> versions. Set this value to an integer.
-purgeBeforeDate		Deletes all versions before <code>purgeBeforeDate</code> . The format for the date must be <code>mm/dd/yyyy</code> .
-purgeFiles		When set to <code>TRUE</code> , transformations and jobs are permanently and physically removed. Note that shared objects (such as database connections) are NOT removed. If you want to also remove shared objects, include the <code>purgeSharedObject</code> parameter as well. If you set the <code>purgeFiles</code> parameter to <code>FALSE</code> , it does not purge files. If you include the <code>purgeFiles</code> parameter in the command, but you don't set it to <code>TRUE</code> or <code>FALSE</code> , the Purge Utility will assume that it is set to <code>TRUE</code> .
-purgeRevisions		When set to <code>TRUE</code> , all revisions are purged, but the current file remains unchanged. If you set the <code>purgeRevisions</code> parameter to <code>FALSE</code> , it does not purge revisions. If you include the <code>purgeRevisions</code> parameter in the command, but you don't set it to <code>TRUE</code> or <code>FALSE</code> , the Purge Utility will assume that it is set to <code>TRUE</code> .
-logFileName	N	Allows you to specify the file name for the log file. If this parameter is not present, the log is written to a file that has this name format: <code>purge-utility-log-YYYYMMdd-HHmmss.txt</code> . <code>YYYYMMdd-HHmmss</code> indicates the date and time that the log file was created (e.g. <code>purge-utility-log-20140313-154741.txt</code> ).
-logLevel	N	Indicates the types and levels of detail the logs should contain. Values are: <code>ALL</code> , <code>DEBUG</code> , <code>ERROR</code> , <code>FATAL</code> , <code>TRACE</code> , <code>INFO</code> , <code>OFF</code> , and <code>WARN</code> . By default the log is set to <code>INFO</code> . Check the Log4j documentation for more details on the logging framework definitions: <a href="https://logging.apache.org/log4j/1.2/apidocs/org/apache/log4j/Level.html">https://logging.apache.org/log4j/1.2/apidocs/org/apache/log4j/Level.html</a> .

- In this example, only the last five revisions of transformations and jobs are NOT deleted. All previous revisions are deleted.

```
purge-utility.bat -url=http://localhost:9080 -user=jdoe -password=mypassword -  
versionCount=5
```

- In the example that follows all revisions before 01/11/2009 are deleted. Logging is set to the WARN level.

```
purge-utility.bat -url=http://localhost:9080 -user=jdoe -password=mypassword -  
purgeBeforeDate=01/11/2009 -logLevel=WARN
```

- In this example, all transformations, jobs, and shared objects are deleted. Note that you do not need to set the `purgeFiles` and `purgeSharedObjects` parameters to `TRUE` for this command to work. Logging is turned OFF.

```
purge-utility.bat -url=http://localhost:9080 -user=jdoe -password=mypassword -  
purgeFiles -purgeSharedObjects -logLevel=OFF
```

1. When finished, examine the logs to see if there were any issues or problems with the purge.
2. To see the results of the purge process, disconnect, then reconnect to the DI Repository. In the **Repository Explorer**, in the **Browse** tab, verify that the items you specified in your purge utility command were purged.

# Set DI Version Control and Comment Tracking Options

By default, PDI tracks versions and comments for jobs, transformations, and connection information when you save them. You can turn version control and comment tracking on or off by modifying the `repository.spring.properties` text file.

## Turn Version Control Off

To turn version control off, complete these steps. If you turn version control off, comment tracking is also turned off.

1. Exit from Spoon.
2. Stop the DI Server.
3. Open the `data-integration-server/pentaho-solution/systems/repository.spring.properties` file in a text editor.
4. Set `versioningEnabled` to `false`.

```
versioningEnabled=false
```

0. 1. Save and close the file.
1. 2. Start the DI Server.
2. 3. Start Spoon.
3. 4. To verify that version control is off, connect to the DI Repository. In the **Repository Explorer** window in the **Browse** tab, click on a file. Note that although you can see the **Access Control** the label, the **Version History** tab is hidden.

## Turn Comment Tracking Off

To turn comment tracking off, complete these steps.

0. 1. Exit from Spoon.
1. 2. Stop the DI Server.
2. 3. Open the `data-integration-server/pentaho-solution/systems/repository.spring.properties` file in a text editor.
3. 4. Set `versioningEnabled` to `true` and `versionCommentsEnabled` to `false`.

```
versioningEnabled=true  
versionCommentsEnabled=false
```

3. 1. Save and close the file.
4. 2. Start the DI Server.
5. 3. Start Spoon.
6. 4. To verify that comment tracking is off, connect to the DI Repository. In the **Repository Explorer** window in the **Browse** tab, click on a file. Note that the **Version History** tab appears, but that the **Comment** field is hidden. Also note that when you save a transformation, job, or connection information, you are no longer prompted to enter a comment.

## Turn Version Control On

To turn version control on, do these things.

1. Exit from Spoon.
2. Stop the DI Server.
3. Open the `data-integration-server/pentaho-solution/systems/repository.spring.properties` file in a text editor.
4. Set `versioningEnabled` to `true`.

```
versioningEnabled=true
```

1. If you want to also turn Comment Tracking, make sure to set `versionCommentsEnabled` to `true`.
2. Save and close the file.
3. Start the DI Server.
4. Start Spoon.
5. To verify that Version Control is on, connect to the DI Repository. In the **Repository Explorer** window in the **Browse** tab, click on a file. Note that the **Version History** tab appears. When you save a transformation, job, or connection information, version control is applied.

## Turn Comment Tracking On

To turn comment tracking on, complete these steps.

1. Exit from Spoon.
2. Stop the DI Server.
3. Open the `data-integration-server/pentaho-solution/systems/repository.spring.properties` file in a text editor.
4. Set `versioningEnabled` to `true` and `versionCommentsEnabled` to `true`.

```
versioningEnabled=true  
versionCommentsEnabled=true
```

1. Save and close the file.

2. Start the DI Server.
3. Start Spoon.
4. To verify that Comment Tracking is on, connect to the DI Repository. In the **Repository Explorer** window in the **Browse** tab, click on a file. Note that the **Version History** tab appears with the **Comments** field. When you save a transformation, job, or connection information, you are prompted to enter a comment.