Pentaho Data Integration Administrator's Guide

This document supports Pentaho Business Analytics Suite 4.8 GA and Pentaho Data Integration 4.4 GA, documentation revision October 31, 2012.

Help and Support Resources

If you have questions that are not covered in this guide, or if you would like to report errors in the documentation, please contact your Pentaho technical support representative.

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

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

For information about instructor-led training on the topics covered in this guide, visit http://www.pentaho.com/training.

Limits of Liability and Disclaimer of Warranty

Trademarks

Company Information

Pentaho Corporation
Citadel International, Suite 340
5950 Hazeltine National Drive
Orlando, FL 32822
Phone: +1 407 812-OPEN (6736)
Fax: +1 407 517-4575
http://www.pentaho.com

E-mail: communityconnection@pentaho.com

Sales Inquiries: sales@pentaho.com

Documentation Suggestions: documentation@pentaho.com

Sign-up for our newsletter: http://community.pentaho.com/newsletter/

Contents

Introduction

This guide contains instructions for configuring and managing a production or development Pentaho Data Integration (PDI) server. **Installation is not covered here**; for installation procedures, refer to the *Pentaho Data Integration Installation Guide* instead.

Adding a JDBC Driver

Before you can connect to a data source in any Pentaho server or client tool, you must first install the appropriate database driver. Your database administrator, Chief Intelligence Officer, or IT manager should be able to provide you with the proper driver JAR. If not, you can download a JDBC driver JAR file from your database vendor or driver developer's Web site. Once you have the JAR, follow the instructions below to copy it to the driver directories for all of the Business Analytics components that need to connect to this data source. See the *Compatibility Matrix: Supported Components* in any of the Installation guide for current version numbers.

Note: Microsoft SQL Server users frequently use an alternative, non-vendor-supported driver called JTDS. If you are adding an MSSQL data source, ensure that you are installing the correct driver.

Backing up old drivers

You must also ensure that there are no other versions of the same vendor's JDBC driver installed in these directories. If there are, you may have to back them up and remove them to avoid confusion and potential class loading problems. This is of particular concern when you are installing a driver JAR for a data source that is the same database type as your Pentaho solution repository. If you have any doubts as to how to proceed, contact your Pentaho support representative for guidance.

Installing JDBC drivers

Copy the driver JAR file to the following directories, depending on which servers and client tools you are using (Dashboard Designer, ad hoc reporting, and Analyzer are all part of the BA Server):

Note: **For the DI Server:** before copying a new JDBC driver, ensure that there is not a different version of the same JAR in the destination directory. If there is, you must remove the old JAR to avoid version conflicts.

- **BA Server:** `/pentaho/server/biserver-ee/tomcat/lib/`
- **Enterprise Console:** `/pentaho/server/enterprise-console/jdbc/`
- **Data Integration Server:** `/pentaho/server/data-integration-server/tomcat/webapps/pentaho-di/WEB-INF/lib/`
- **Data Integration client:** `/pentaho/design-tools/data-integration/libext/JDBC/`
- **Report Designer:** `/pentaho/design-tools/report-designer/lib/jdbc/`
- **Schema Workbench:** `/pentaho/design-tools/schema-workbench/drivers/`
- **Aggregation Designer:** `/pentaho/design-tools/agg-designer/drivers/`
- **Metadata Editor:** `/pentaho/design-tools/metadata-editor/libext/JDBC/`

Note: To establish a data source in the Pentaho Enterprise Console, you must install the driver in both the Enterprise Console and the BA Server or Data Integration Server. If you are just adding a data source through the Pentaho User Console, you do not need to install the driver to Enterprise Console.

Restarting

Once the driver JAR is in place, you must restart the server or client tool that you added it to.

Connecting to a Microsoft SQL Server using Integrated or Windows Authentication

The JDBC driver supports Type 2 integrated authentication on Windows operating systems through the **integratedSecurity** connection string property. To use integrated authentication, copy the **sqljdbc_auth.dll** file to all the directories to which you copied the JDBC files.

The **sqljdbc_auth.dll** files are installed in the following location:

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

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

PDI Functions in the Pentaho Enterprise Console

The PDI features of the Pentaho Enterprise Console allow you to monitor all activity on a Data Integration server remotely. You can also register specific jobs and transformations to monitor for more detailed information such as performance trends, execution history, and error logs.

Important: Before you can monitor jobs and transformations, you must first configure logging and monitoring features in Pentaho Data Integration. See *How to Enable Logging* on page 33 for details.

To start viewing jobs and transformations, you must perform at least one of the following tasks:

- Configure the connection to the Data Integration Server you want to monitor
- Register transformations and jobs specific transformations and jobs you want to monitor in detail

Connecting to the Data Integration Server

You must register your Data Integration server to link it to the Pentaho Enterprise Console. After registration you can monitor activity associated with jobs and transformations being processed by the Data Integration server from the PDI dashboard in the Pentaho Enterprise Console.

To register your Data Integration server:

1. Make sure your Data Integration server is up and running.
2. In the Pentaho Enterprise Console home page, click **Pentaho Data Integration**.
3. Click to open the **Carte Configuration** dialog box.
4. In the **Carte Configuration** dialog box, enter the Data Integration server URL, (for example, http://localhost:9080/pentaho-di/), the server user name, and password.

   Note: You can still configure the Pentaho Enterprise Console to connect to a basic Carte server using the appropriate URL; (for example, http://localhost:80/kettle/).
5. Click **OK** to complete the registration.
   A message appears if the connection to the Data Integration server is successful. If you see an error message, check to make sure your Data Integration server is running and that your access credentials are correct.

Monitoring Current Activity and Alerts

Once your Data Integration or Carte server is running and you have registered jobs and transformations, the **Current Activity** chart on the PDI dashboard displays the total number of jobs and transformations that are waiting, running, or paused on a configured server. Also displayed is a list of alerts associated with registered jobs and transformations. Alerts help you determine if a job or transformation is taking too long to run, has completed too quickly, or has logged errors.

Click **Refresh** to update the dashboard. Click in the **History** list box to view performance history (All History) associated with registered jobs and transformations.

Registering PDI Jobs and Transformations

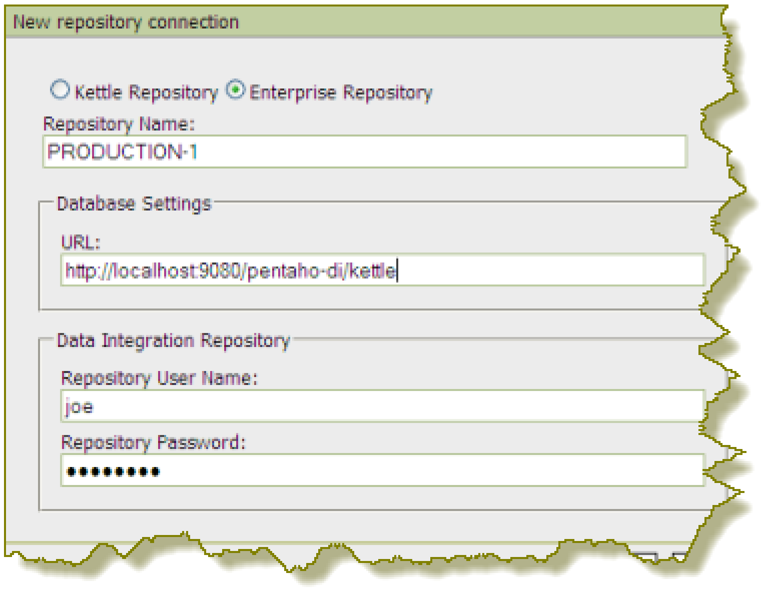
The Pentaho Enterprise Console allows you to register jobs and transformations contained in an enterprise or database repository. Registering a job or transformation allows you to analyze additional related information such as the log history and performance trends.

Registering Transformations and Jobs from the Pentaho Enterprise Repository

To register jobs and transformations stored in a database repository, you must connect to the repository by defining a database connection.

1. Make sure your Data Integration server is running.
2. In the Pentaho Enterprise Console, click the **Pentaho Data Integration** tab.

3. Click (Registration) to view the **Register Jobs and Transformations** page.
4. In the **Register Jobs and Transformations** page, click the plus sign (+) next to **Repository**.
   The **New Repository Connection** dialog box appears.
5. Enter the information about your repository in the appropriate fields and click **OK.**


New repository connection

Kettle Repository [✓ Enterprise Repository]

Repository Name: PRODUCTION-1

Database Settings

URL: http://localhost:9080/pentaho-di/kettle

Data Integration Repository

Repository User Name: joe

Repository Password: ••••••••

Note: No fields in the **New Repository Connection** dialog box are allowed to be blank and your port number must be valid.

6. Select the new repository and click **Browse.**
   When you connect to your database repository successfully, a list of all jobs and transformations in the repository appears.
7. Select the jobs and transformations you want to monitor and click **Register**.

   Note: Use **CTRL+ CLICK** to select multiple jobs and transformations.

   The jobs and transformations you selected appear in the **Registered** list box. Previously registered jobs and transformations are not displayed.

Registering Transformations and Jobs from a Database Repository

To register jobs and transformations stored in a classic Kettle database repository, you must connect to the repository by defining a database connection.

Caution: To avoid database connection errors, be sure you have accurate database connection details. Depending on the database vendor, incorrect entries associated with database connections result in error messages that may not identify issues clearly.

1. Make sure your repository is running.
2. In the Pentaho Enterprise Console, click the **Pentaho Data Integration** tab.
3. Click (Registration) to view the **Register Jobs and Transformations** page.
4. In the **Register Jobs and Transformations** page, click the plus sign (+) next to **Repository**.
   The **New Repository Connection** dialog box appears.
5. Click the **Kettle Repository** radio button.
6. Enter the information about your database in the appropriate fields and click **OK.**

   Note: No fields in the **New Repository Connection** dialog box are allowed to be blank and your port number must be valid.

7. Select the new repository and click **Browse.**

When you connect to your database repository successfully, a list of all jobs and transformations in the database repository appears.

8. Select the jobs and transformations you want to monitor and click **Register**.

   Note: Use **CTRL+ CLICK** to select multiple jobs and transformations.

   The jobs and transformations you selected appear in the **Registered** list box.

Registering Transformations and Jobs from a File System

To register and view jobs and transformations that are stored in a file system, you must browse for the files that contain them.

1. In the Pentaho Enterprise Console, click the **Pentaho Data Integration** tab.
2. Click ✓ (Registration) to view the **Register Jobs and Transformations** page.
3. In the **File** text box type the path name to the file that contains the job or transformation or click **Browse** to locate the file.
4. Click **Register**
   The jobs and transformations you selected appear in the **Registered** list box.

Monitoring Jobs and Transformations

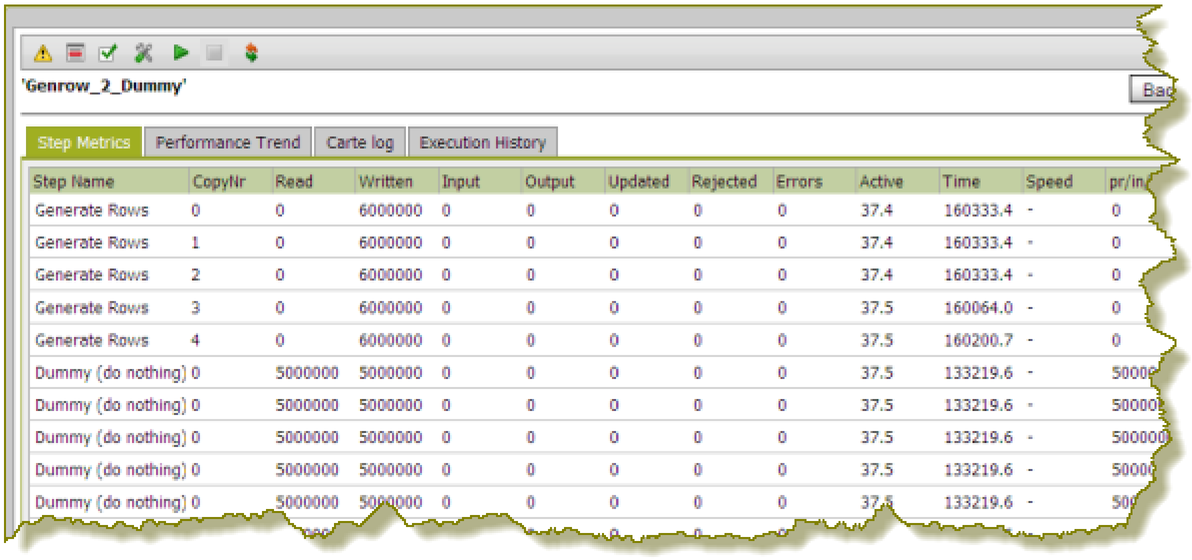
You can monitor the status of all PDI-related jobs and transformations from the Pentaho Enterprise Console. In the console, click the PDI menu item then click (Monitoring Status) to display all jobs and transformations that are currently running on the Carte server and those which are registered. You will see all registered jobs and transformations and all jobs and transformations that have run on Carte since it was last restarted. Registered jobs and transformations that are registered but which have not been run on Carte since it was last restarted, display a status of "unpublished." Also displayed are runtime thresholds (Alert Min and Max), if specified.

From this page you can start running ▶ pause ⏸, and stop running ■ jobs and transformations that have been sent to the server since it was last restarted. Click (Refresh) to refresh the list of jobs and transformations as needed.

Monitoring Performance Trends for Jobs and Transformations

The job and transformation details page associated with the PDI feature of the Pentaho Enterprise Console allows you, among other things, to look a performance trends for a specific job or transformation, to set up alert thresholds, and render results by occurrence or date. In addition, you can view the job or transformation log file (if running on the Data Integration server or Carte) and the step metrics (if running on the Data Integration server or Carte) associated with a specific job or transformation.

To display the job and transformation details page, click the PDI tab in the console, then click (Monitoring Status) to open the Monitor Status page. Double-click on a specific job and transformation to display the activity details page. The information available on the details page depends on the settings you configured in the Transformation Settings dialog box in Spoon. See the *Pentaho Data Integration User Guide* for details.



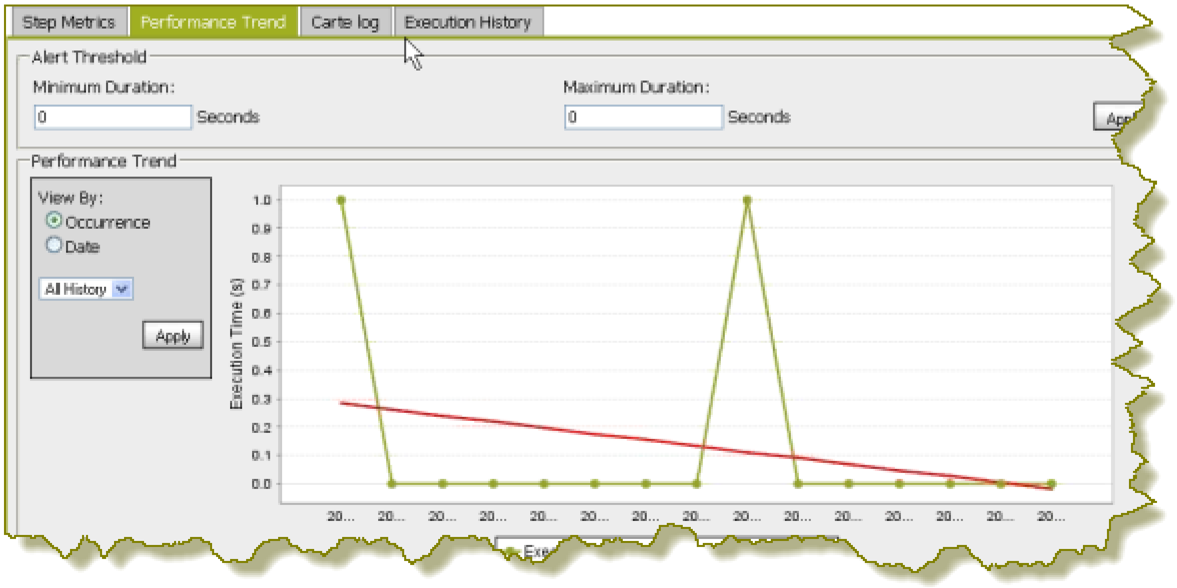
'Genrow_2_Dummy'

Step Metrics | Performance Trend | Carte log | Execution History

Step Name	CopyNr	Read	Written	Input	Output	Updated	Rejected	Errors	Active	Time	Speed	pr/in
Generate Rows	0	0	6000000	0	0	0	0	0	37.4	160333.4	-	0
Generate Rows	1	0	6000000	0	0	0	0	0	37.4	160333.4	-	0
Generate Rows	2	0	6000000	0	0	0	0	0	37.4	160333.4	-	0
Generate Rows	3	0	6000000	0	0	0	0	0	37.5	160064.0	-	0
Generate Rows	4	0	6000000	0	0	0	0	0	37.5	160200.7	-	0
Dummy (do nothing)	0	5000000	5000000	0	0	0	0	0	37.5	133219.6	-	5000
Dummy (do nothing)	0	5000000	5000000	0	0	0	0	0	37.5	133219.6	-	50000
Dummy (do nothing)	0	5000000	5000000	0	0	0	0	0	37.5	133219.6	-	500000
Dummy (do nothing)	0	5000000	5000000	0	0	0	0	0	37.5	133219.6	-	5000
Dummy (do nothing)	0	5000000	5000000	0	0	0	0	0	37.5	133219.6	-	50

Step Metrics displays metrics at a step level associated with how many rows each step has processed (read, written, input, output.)

The **Performance Trend** chart displays the performance of a job or transformation over time. Performance history is displayed for registered jobs and transformations that are configured with database logging exclusively.



Alert Threshold

Minimum Duration: 0 Seconds

Maximum Duration: 0 Seconds

Performance Trend

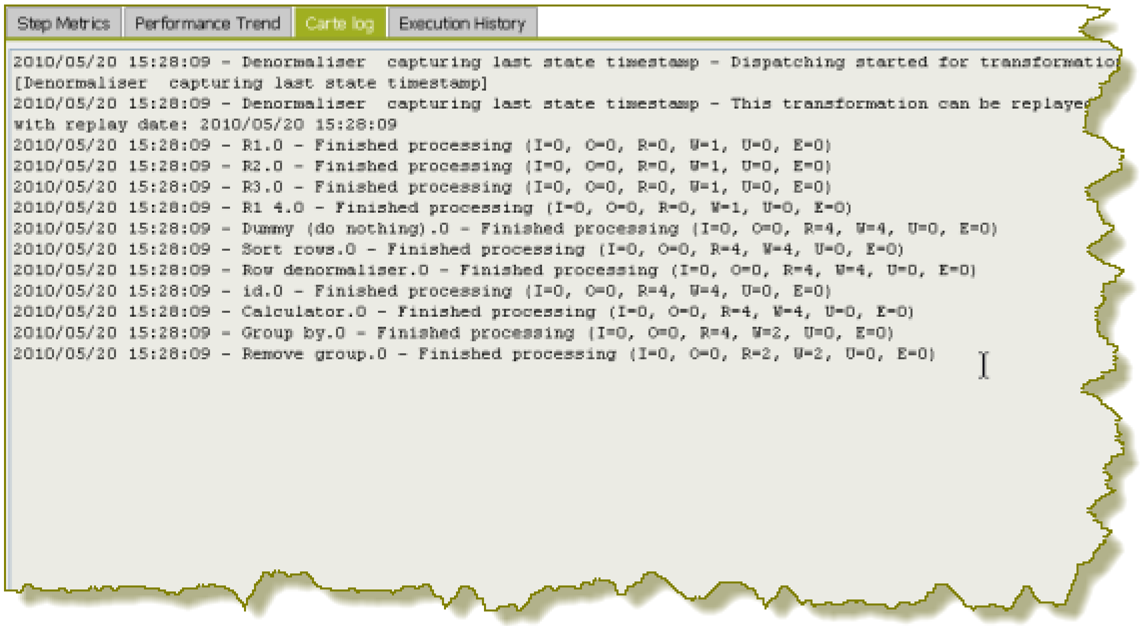
View By: [✓ Occurrence] Date

All History

Note: The Performance Trend is viewable only if you configured the transformation to log to a database in Spoon. See the *Pentaho Data Integration User Guide* for details.

You can configure an **Alert Threshold** by editing the runtime durations in number of seconds. Click **Apply** to have changes take effect. A line appears on the performance chart that displays the minimum and maximum durations. These values are also used on the status page to display warning messages when applicable. The Performance Trend chart can be adjusted to render results by date or by occurrence. For each of these chart types, a filter can be applied to limit the results. Click **Apply** to have changes take effect.

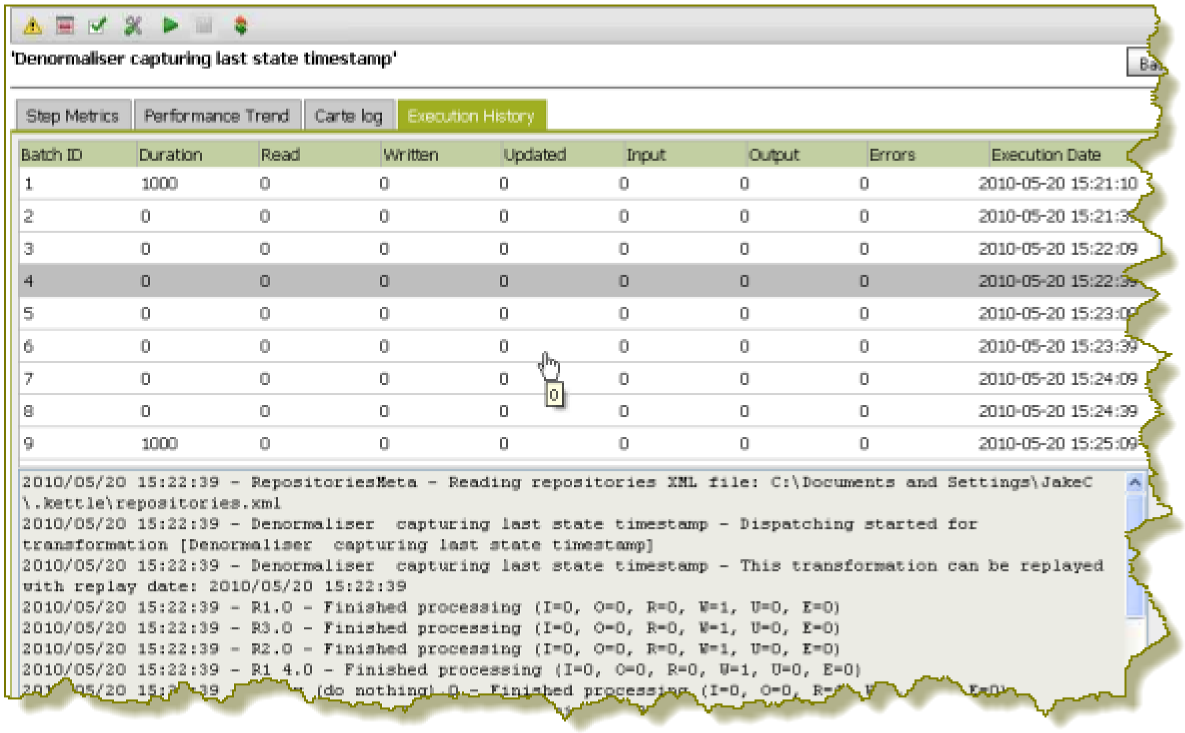
Carte Log displays all relevant logging information associated with the execution of the transformation.



```
2010/05/20 15:28:09 - Denormaliser  capturing last state timestamp - Dispatching started for transformatio
[Denormaliser  capturing last state timestamp]
2010/05/20 15:28:09 - Denormaliser  capturing last state timestamp - This transformation can be replaye
with replay date: 2010/05/20 15:28:09
2010/05/20 15:28:09 - R1.0 - Finished processing (I=0, O=0, R=0, W=1, U=0, E=0)
2010/05/20 15:28:09 - R2.0 - Finished processing (I=0, O=0, R=0, W=1, U=0, E=0)
2010/05/20 15:28:09 - R3.0 - Finished processing (I=0, O=0, R=0, W=1, U=0, E=0)
2010/05/20 15:28:09 - R1 4.0 - Finished processing (I=0, O=0, R=0, W=1, U=0, E=0)
2010/05/20 15:28:09 - Dummy (do nothing).0 - Finished processing (I=0, O=0, R=4, W=4, U=0, E=0)
2010/05/20 15:28:09 - Sort rows.0 - Finished processing (I=0, O=0, R=4, W=4, U=0, E=0)
2010/05/20 15:28:09 - Row denormaliser.0 - Finished processing (I=0, O=0, R=4, W=4, U=0, E=0)
2010/05/20 15:28:09 - id.0 - Finished processing (I=0, O=0, R=4, W=4, U=0, E=0)
2010/05/20 15:28:09 - Calculator.0 - Finished processing (I=0, O=0, R=4, W=4, U=0, E=0)
2010/05/20 15:28:09 - Group by.0 - Finished processing (I=0, O=0, R=4, W=2, U=0, E=0)
2010/05/20 15:28:09 - Remove group.0 - Finished processing (I=0, O=0, R=2, W=2, U=0, E=0)
```

Execution History allows you to view log information from previous executions. This is particularly helpful when you are troubleshooting an error. In the image below, the transformation has run multiple times. The user has selected to display details about the fourth time the transformation ran.

Note: Execution History is viewable only if you configured the transformation to log to a database in Spoon (LOG_FIELD table). See the *Pentaho Data Integration User Guide* for details.



Batch ID	Duration	Read	Written	Updated	Input	Output	Errors	Execution Date
1	1000	0	0	0	0	0	0	2010-05-20 15:21:10
2	0	0	0	0	0	0	0	2010-05-20 15:21:3
3	0	0	0	0	0	0	0	2010-05-20 15:22:09
4	0	0	0	0	0	0	0	2010-05-20 15:22:3
5	0	0	0	0	0	0	0	2010-05-20 15:23:0
6	0	0	0	0	0	0	0	2010-05-20 15:23:39
7	0	0	0	0	0	0	0	2010-05-20 15:24:09
8	0	0	0	0	0	0	0	2010-05-20 15:24:39
9	1000	0	0	0	0	0	0	2010-05-20 15:25:09

```
2010/05/20 15:22:39 - RepositoriesMeta - Reading repositories XML file: C:\Documents and Settings\JakeC
\.kettle\repositories.xml
2010/05/20 15:22:39 - Denormaliser  capturing last state timestamp - Dispatching started for
transformation [Denormaliser  capturing last state timestamp]
2010/05/20 15:22:39 - Denormaliser  capturing last state timestamp - This transformation can be replayed
with replay date: 2010/05/20 15:22:39
2010/05/20 15:22:39 - R1.0 - Finished processing (I=0, O=0, R=0, W=1, U=0, E=0)
2010/05/20 15:22:39 - R3.0 - Finished processing (I=0, O=0, R=0, W=1, U=0, E=0)
2010/05/20 15:22:39 - R2.0 - Finished processing (I=0, O=0, R=0, W=1, U=0, E=0)
2010/05/20 15:22:39 - R1 4.0 - Finished processing (I=0, O=0, R=0, W=1, U=0, E=0)
201 05/20 15:2 :39 (do nothing).0 - Finished processing (I=0, O=0, R= E=0)
```

Installing or Updating License Keys Using the Pentaho Enterprise Console

1. Copy your LIC files to the server that runs the Pentaho Enterprise Console.
2. Log into the Pentaho Enterprise Console by opening a web browser and navigating to `www.<serverHostName:8088`, changing <serverHostName> to the host name or IP address of your BA or DI server.
3. Click the + (plus) button in the upper-right corner of the **Subscriptions** section. **Browse**, then navigate to your LIC files and open them.
   The Install License dialog box appears.
4. Click Browse and navigate to the LIC files you want to install and open them.
5. After opening all LIC files, click **OK**.

You can configure your licensed products through the Pentaho Enterprise Console.

Working From the Command Line Interface

Though the Pentaho Enterprise Console is the quickest, easiest, and most comprehensive way to manage PDI and/or the BA Server, some Pentaho customers may be in environments where it is difficult or impossible to deploy or use the console. See the alternative instructions for command line interface (CLI) license registration for step-by-step instructions.

Installing License Keys from the Command-line Interface

1. Download the related archive package for the Pentaho product LIC file you want to install.
2. Copy your LIC files to the server that runs the Pentaho Enterprise Console.
3. Navigate to the `/pentaho/server/enterprise-console/license-installer/` directory, or the `/license-installer/` directory that was part of the archive package you downloaded.
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/pgibbons/downloads/Pentaho\ BI\ Platform\ 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\pgibbons\Downloads\Pentaho BA Platform Enterprise Edition.lic"

Security and Authorization Configuration

The information in this section explains how to configure users, roles, and other permissions settings for your PDI enterprise repository.

Changing the Admin Credentials for the Pentaho Enterprise Console

The default user name and password for the Pentaho Enterprise Console are **admin** and **password**, respectively. You must change these credentials if you are deploying the BA Server to a production environment. Follow the instructions below to change the credentials:

1. Stop the Pentaho Enterprise Console.

   ```
   /pentaho/server/enterprise-console/stop-pec.sh
   ```
2. Open a terminal or command prompt window and navigate to the `/pentaho/server/enterprise-console/` directory.
3. Execute the **pec-passwd** script with two parameters: the Enterprise Console username and the new password you want to set.

   This script will make some configuration changes for you, and output an obfuscated password hash to the terminal.

   ```
   ./pec-passwd.sh admin newpass
   ```
4. Copy the hash output to the clipboard, text buffer, or to a temporary text file.
5. Edit the `/pentaho/server/enterprise-console/resource/config/login.properties` file with a text editor.
6. Replace the existing hash string with the new one, leaving all other information in the file intact.

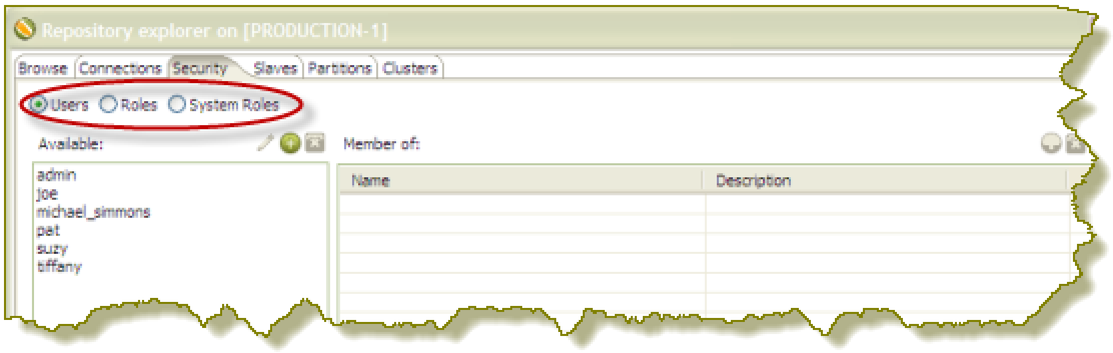
   ```
   admin: OBF:1uo91vn61ymf1yt41v1p1ym71v2p1yti1ylz1vnw1unp,server-
   administrator,content-administrator,admin
   ```
7. Save and close the file, and restart Enterprise Console.

The Pentaho Enterprise Console password is now changed.

Managing Users and Roles in the Pentaho Enterprise Repository

Pentaho Data Integration comes with a default security provider. If you don't have an existing authentication provider such as LDAP or MSAD, you can use Pentaho Security to define users and roles.

The point-and-click user interface for users and roles in the Pentaho Enterprise Repository similar to the one in the Pentaho Enterprise Console. The users and roles radio buttons allow you to switch between user and role settings. You can add, delete, and edit users and roles from this page.



admin
joe
michael_simmons
pat
suzy
tiffany

Adding Users

You must be logged into the Enterprise Repository as an administrative user.

To add users in the Enterprise Repository, follow the directions below.

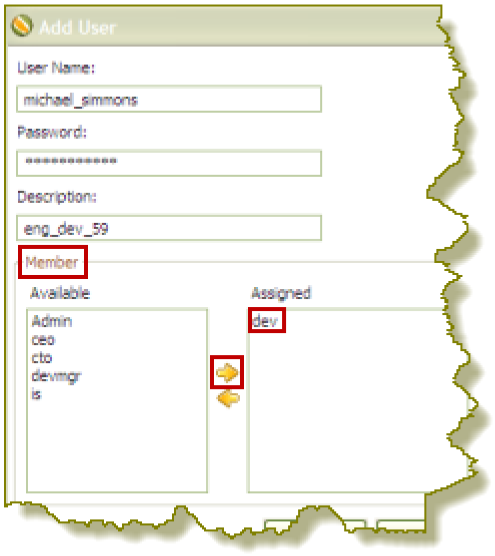
1. In Spoon, go to **Tools** -> **Repository** -> **Explore**.
   The Repository Explorer opens.
2. Click the **Security** tab.

   Note: The **Users** radio button is selected by default.

3. Next to **Available**, click (**Add**).
   The **Add User** dialog box appears.
4. Enter the **User Name** and **Password** associated with your new user account in the appropriate fields.

   Note: An entry in the **Description** field is optional.

5. If you have available roles that can be assigned to the new user, under **Member**, select a role and click (**Add**).



   The role you assigned to the user appears in the right pane under **Assigned**.
6. Click **OK** to save your new user account and exit the Add Users dialog box.

The name of the user you added appears in the list of Available users.

Editing User Information

You must be logged into the Enterprise Repository as an administrative user.

Follow the instructions below to edit a user account.

1. In Spoon, go to **Tools** -> **Repository** -> **Explore**.
   The Repository Explorer opens.
2. Click the **Security** tab.

   Note: The **Users** radio button is selected by default.

3. Select the user whose details you want to edit from the list of available users.
4. Click (Edit)
   The **Edit User** dialog box appears.
5. Make the appropriate changes to the user information.
6. Click **OK** to save changes and exit the Edit User dialog box.

Deleting Users

You must be logged into the Enterprise Repository as an administrative user. Refer to *Best Practices for Deleting Users and Roles in the Pentaho Enterprise Repository* before you delete a user or role.

Follow the instructions below to delete a user account:

1. In Spoon, go to **Tools** -> **Repository** -> **Explore**.
   The Repository Explorer opens.
2. Click the **Security** tab.
3. Select the user you want to delete from the list of available users.
4. Next to **Users**, click (**Remove**).
   A confirmation message appears.
5. Click **Yes** to delete the user.

The specified user account is deleted.

Best Practices for Deleting Users and Roles in the Pentaho Enterprise Repository

If a user or role is deleted in the Pentaho Enterprise Repository, (currently used by Pentaho Data Integration), content that refers to the deleted user (either by way of owning the content or having an ACL that mentions the user or role) is left unchanged; therefore, it is possible that you may create a new user or role, at a later date, using an identical name. In this scenario, content ownership and access control entries referring to the deleted user or role now apply to the new user or role.

To avoid this problem, Pentaho recommends that you disable a user or role instead of deleting it. This prevents a user or role with an identical name from ever being created again. The departmental solution (also referred to as Hibernate or the Pentaho Security back-end), does not have disable functionality. For this back-end, Pentaho recommends that you use the following alternatives rather than deleting the user or role:

IF	THEN
You are dealing with a role	Unassign all current members associated with the role
You are dealing with a user	Reset the password to a password that is so cryptic that it is impossible to guess and is unknown to any users

Adding Roles

You must be logged into the Enterprise Repository as an administrative user.

To add roles in the Enterprise Repository, follow the directions below:

1. In Spoon, go to **Tools** -> **Repository** -> **Explore**.
   The Repository Explorer opens.
2. Click the **Security** tab.
3. Click the **Roles** radio button.
   The list of available roles appear.
4. Click (Add)
   The **Add Role** dialog box appears.
5. Enter the **Role Name** in the appropriate field.

   Note: An entry in the **Description** field is optional.

6. If you have users to assign to the new role, select them (using the <**SHIFT**> or <**CTRL**> keys) from the list of available users and click (**Add**).
   The user(s) assigned to your new role appear in the right pane.
7. Click **OK** to save your entries and exit the Add Role dialog box.

The specified role is created and is ready to be assigned to user accounts.

Editing Roles

You must be logged into the Enterprise Repository as an administrative user.

To edit roles in the Enterprise Repository, follow the directions below.

1. In Spoon, go to **Tools** -> **Repository** -> **Explore**.
   The Repository Explorer opens.

2. Click the **Security** tab.
3. Click the **Roles** radio button.
   The list of available roles appear.
4. Select the role you want to edit and click (**Edit**)
   The **Edit Role** dialog box appears.
5. Make the appropriate changes.
6. Click **OK** to save your changes and exit the **Edit Role** dialog box.

Deleting Roles

You must be logged into the Enterprise Repository as an administrative user. Refer to *Best Practices for Deleting Users and Roles in the Pentaho Enterprise Repository* before you delete a user or role.

Follow the instructions below to delete a role:

1. In Spoon, go to **Tools** -> **Repository** -> **Explore**.
   The Repository Explorer opens.
2. Click the **Security** tab.
3. Select the role you want to delete from the list of available roles.
4. Click (**Remove**).
   A confirmation message appears.
5. Click **Yes** to delete the role.

The specified role is deleted.

Assigning Users to Roles

You must be logged into the Enterprise Repository as an administrative user.

You can assign users to roles, (and vice-versa), when you add a new user or role; however, you can also assign users to roles as a separate task.

1. In Spoon, go to **Tools** -> **Repository** -> **Explore**.
   The Repository Explorer opens.
2. Click the **Security** tab.
3. Click the **Roles** radio button.
   The list of available roles appear.
4. Select the role to which you want to assign one or more users.

   Note: If the role has users currently assigned to it, the names of the users appear in the table on the right under **Members**. You can assign or unassign any users to a role. You can select a single item or multiple items from the list of members. Click (Remove) to remove the user assignment.
5. Next to **Members**, click (**Add**).
   The **Add User to Role** dialog box appears.
6. Select the user (or users) you want assigned to the role and click (**Add**).
   The user(s) assigned to the role appear in the right pane.
7. Click **OK** to save your entries and to exit the Add User to Role dialog box.

The specified users are now assigned to the specified role.

Making Changes to the Admin Role

The assigning of task-related permissions, (Read, Create, and Administrate), associated with the Admin role in the Pentaho Enterprise Repository cannot be edited in the user interface. The Admin role is the only role that is assigned the *Administrate* permission; the Administrate permission controls user access to the Security tab.

Deleting the Admin role prevents *all users* from accessing the Security tab, unless another role is assigned the Administrate permission. Below are the scenarios that require a non-UI configuration change:

- You want to delete the Admin role
- You want to unassign the Administrate permission from the Admin role

- You want to setup LDAP

Follow the instructions below to change the Admin role:

1. Shut down the Data Integration server.
2. Open the **repository.spring.xml** file located at `\pentaho\server\data-integration-server\pentaho-solutions\system\`.
3. Locate the element with an ID of **immutableRoleBindingMap**.
4. Replace the entire node with the XML shown below. Make sure you change **yourAdminRole** to the role that will have **Administrate** permission.

```
<util:map id="immutableRoleBindingMap">
    <entry key="yourAdminRole">
      <util:list>
        <value>org.pentaho.di.reader</value>
        <value>org.pentaho.di.creator</value>
        <value>org.pentaho.di.securityAdministrator</value>
      </util:list>
    </entry>
</util:map>
```

5. Restart the Data Integration server.

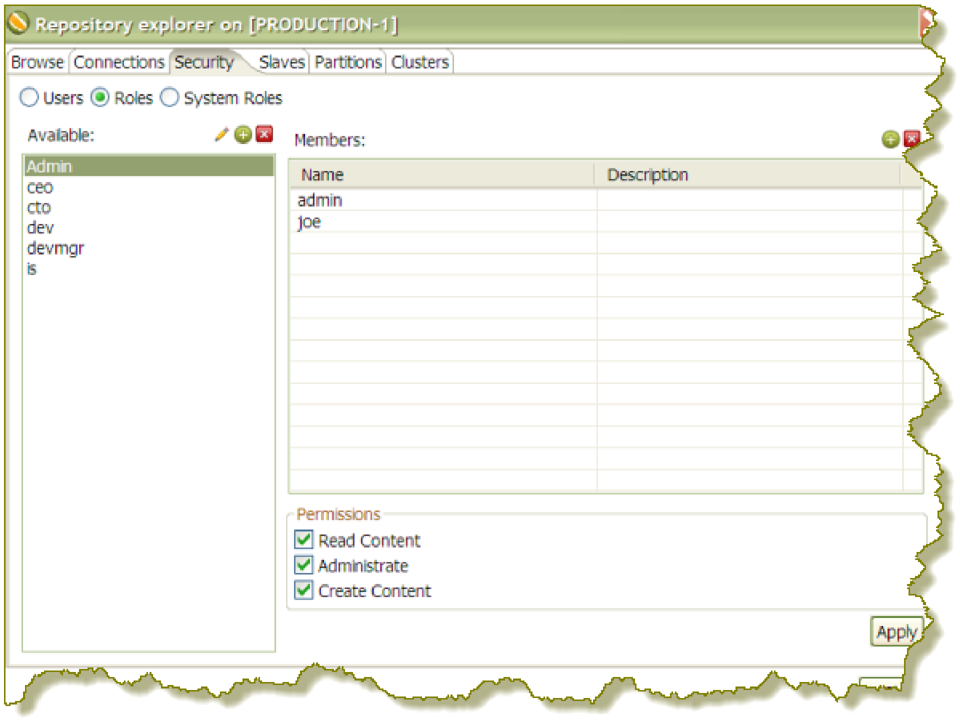
The Admin role is changed according to your requirements.

Assigning Permissions in the Pentaho Enterprise Repository

You must be logged into the Enterprise Repository as an administrative user.

There are "action based" permissions associated with the roles. Roles help you define what users or members of a group have the permission to do. You can create roles that restrict users to reading content exclusively. You can create administrative groups who are allowed to administer security and create new content.

In the example below, user "joe" has an "admin" role. As such, Joe's permissions allow him to **Read Content**, **Administer Security**, and **Create Content**.



Admin
ceo
cto
dev
devmgr
is

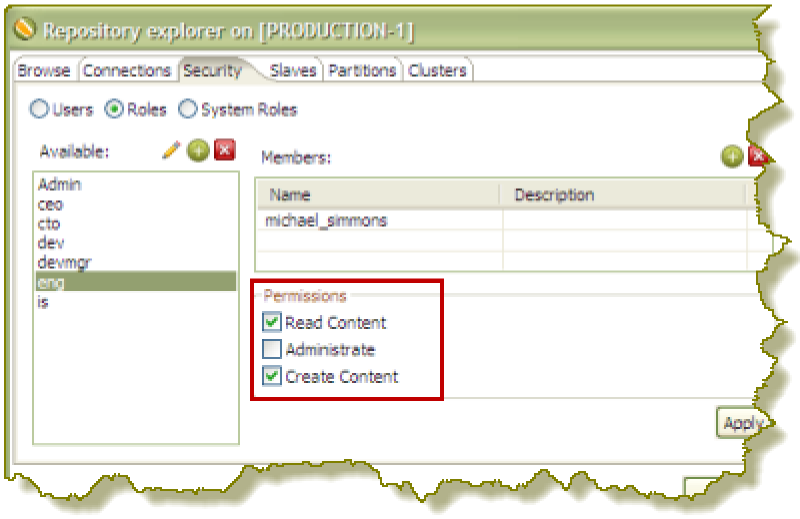
Name	Description
admin	
joe	

Permissions
- [x] Read Content
- [x] Administrate
- [x] Create Content

To assign permissions in the Enterprise Repository, follow the instructions below.

1. In Spoon, go to **Tools** -> **Repository** -> **Explore**.
   The Repository Explorer opens.

2. Click the **Security** tab.
3. Click the **Roles** radio button.
   The list of available roles appear.
4. Select the role to which you want to assign permissions.
5. Enable the appropriate permissions for your role as shown in the example below.



Admin
ceo
cto
dev
devmgr
eng
is

michael_simmons

Permissions
[✓] Read Content
[ ] Administrate
[✓] Create Content

6. Click **Apply**.

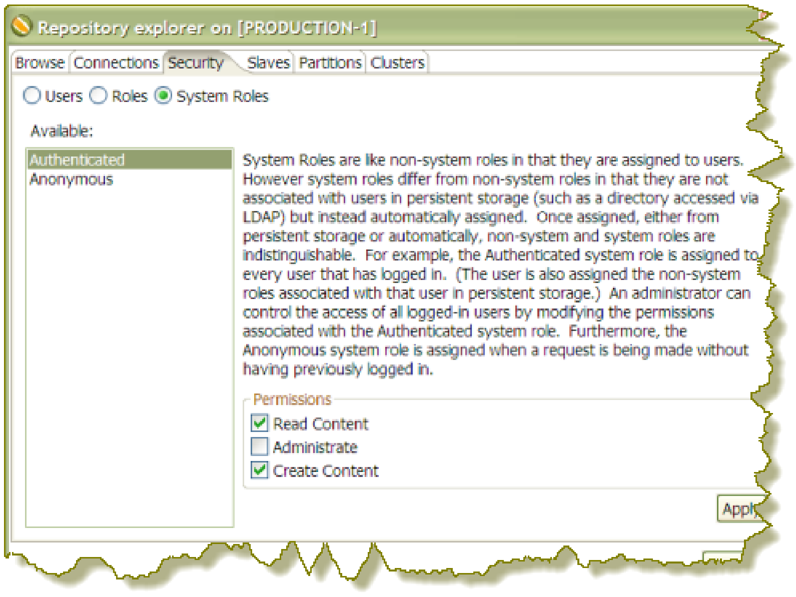
The permissions you enabled for the role will take effect the next time the specified user(s) log into the Pentaho Enterprise Console.

Permissions Settings

Permission	Effect
Read Content	Allows a user or role to examine contents (for example, transformations, jobs, and database connections) in the repository
Administrate	Assigns all permissions to the specified user or role
Create Content	Allows a user or role to create content in the repository

Enabling System Role Permissions in the Pentaho Enterprise Repository

When users log into the Pentaho Enterprise Repository, they are automatically assigned the **Authenticated** system role in addition to the role you assigned to them. Pentaho requires the Authenticated system role for users to log into the Pentaho Enterprise Repository; this includes Admin users. By default, the Authenticated system role provide **Read Content** and **Create Content** permissions to all users who are logged in. You can change these permissions as needed.



System Roles are like non-system roles in that they are assigned to users. However system roles differ from non-system roles in that they are not associated with users in persistent storage (such as a directory accessed via LDAP) but instead automatically assigned. Once assigned, either from persistent storage or automatically, non-system and system roles are indistinguishable. For example, the Authenticated system role is assigned to every user that has logged in. (The user is also assigned the non-system roles associated with that user in persistent storage.) An administrator can control the access of all logged-in users by modifying the permissions associated with the Authenticated system role. Furthermore, the Anonymous system role is assigned when a request is being made without having previously logged in.

Permissions
- [✓] Read Content
- [ ] Administrate
- [✓] Create Content

Note: **Important!** The Anonymous system role is not being used at this time.

Follow the steps below to change permissions for the Authenticated system role.

1. In Spoon, go to **Tools** -> **Repository** -> **Explore**.
   The Repository Explorer opens
2. Click the **Security** tab.
3. Click the **System Roles** radio button.
   The list of available system roles appear.

   Note: The Anonymous role is not functional.

4. Select the **Authenticated** role from the list of available roles.
5. Under **Permissions**, enable the appropriate *permissions* for this role.
6. Click **Apply** to save your changes.

The specified permissions are enabled for the Authenticated system role.

Configuring the DI Server for LDAP Authentication

Your directory server must be available in order to perform this procedure.

Follow the instructions below to configure your DI Server to authenticate against an LDAP service.

Note: **Important!** LDAP-related configuration options in the Pentaho Enterprise Console are strictly for BA Server support and cannot be used to manage LDAP for the Pentaho Data Integration server.

1. Stop the DI Server.
2. Edit the `/pentaho-solutions/system/pentaho-spring-beans.xml` file and replace **hibernate** with **ldap** in both lines.

```
<import resource="applicationContext-spring-security-hibernate.xml" />
<import resource="applicationContext-pentaho-security-hibernate.xml" />
```

3. Save and close the file, then edit the following files in the `/pentaho/server/biserver-ee/pentaho-solutions/system/` directory and change all instances of the **Admin** and **Authenticated** role values to match the appropriate roles in your LDAP configuration:

   - pentaho.xml
   - repository.spring.xml
   - applicationContext-spring-security.xml

4. Start the Data Integration server.

You are now running the Pentaho Data Integration Server in LDAP mode, though you will need to consult *LDAP Properties* on page 20 for more information on LDAP and MSAD configuration settings.

LDAP Properties

You can configure LDAP values by editing the `/pentaho-solutions/system/applicationContext-security-ldap.properties` file in your BA Server or DI Server directory, or through the Pentaho Enterprise Console for the BA Server (the LDAP options in PEC apply only to the BA Server, not the DI Server).

Connection Information (Context)

These entries define connections involving LDAP users (typically administrators) that can execute directory searches.

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

Users

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

Note: The **{0}** token will be replaced by the username from the login dialogue.

Note: The example above defines **DC=Valyant,DC=local** in **contextSource.providerURL**. Given that definition, you would not need to repeat that in **userSearch.searchBase** below because it will be appended automatically to the defined value here.

LDAP Property	Purpose	Example
userSearch.searchBase	Base (by username) for user searches	userSearch.searchBase=CN=Users
userSearch.searchFilter	Filter (by username) for user searches. The attribute you specify here must contain the value that you want your users to log into Pentaho with. Active Directory usernames are represented by **sAMAccountName**; full names are represented by **displayName**.	userSearch.searchFilter=(sAMAccountName=

Populator

The populator matches fully distinguished user names from **userSearch** to distinguished role names for roles those users belong to.

Note: The {0} token will be replaced with the user DN found during a user search; the {1} token is replaced with the username entered in the login screen.

LDAP Property	Purpose	Example
populator.convertToUpperCase	Indicates whether or not retrieved role names are converted to uppercase	populator.convertToUpperCase=false
populator.groupRoleAttribute	The attribute to get role names from	populator.groupRoleAttribute=cn

LDAP Property	Purpose	Example
populator.groupSearchBase	Base (by user DN or username) for role searches.	populator.groupSearchBase=ou=Pentaho
populator.groupSearchFilter	The special nested group filter for Active Directory is shown in the example; this will not work with non-MSAD directory servers.	populator.groupSearchFilter=(memberof:1.2.8
populator.rolePrefix	A prefix to add to the beginning of the role name found in the group role attribute; the value can be an empty string.	populator.rolePrefix=
populator.searchSubtree	Indicates whether or not the search must include the current object and all children. If set to **false**, the search must include the current object only.	populator.searchSubtree=true

All Authorites Search

These entries populate the BI Platform Access Control List (ACL) roles. These should be similar or identical to the Populator entries.

LDAP Property	Purpose	Example
allAuthoritiesSearch.roleAttribute	The attribute used for role values	allAuthoritiesSearch.roleAttribute=cn
allAuthoritiesSearch.searchBase	Base for "all roles" searches	allAuthoritiesSearch.searchBase=ou=Pentaho
allAuthoritiesSearch.searchFilter	Filter for "all roles" searches. Active Directory requires that the **objectClass** value be set to **group**.	allAuthoritiesSearch.searchFilter=(objectClass

All user name search

These entries populate the BI Platform ACL users.

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

Clustering

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.

Configuring 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 an enterprise 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 enterprise repository to retrieve PDI 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.

See the *Pentaho Data Integration User Guide* for more information about assigning slaves and configuring clusters.

Configuring 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.

Configuring 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 *Executing Scheduled Jobs on a Remote Carte Server* on page 26, 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 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 master to use in a dynamic cluster. You must configure one or more Carte slave servers in order for this to be useful.

See the *Pentaho Data Integration User Guide* for more information about configuring clusters in Spoon.

Configuring 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 an enterprise 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 enterprise 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.

See the *Pentaho Data Integration User Guide* for more information about assigning slaves and configuring clusters in Spoon.

Creating 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

Option	Description
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 **Select slave servers** to select from the list of available slave servers.

Executing 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

Initializing 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

Option	Description
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.

Executing 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 enterprise 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>TrustedIpAddrs</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.

Impact Analysis

To see what effect your transformation will have on the data sources it includes, go to the **Action** menu and click on **Impact**. PDI will perform an impact analysis to determine how your data sources will be affected by the transformation if it is completed successfully.

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 *How to Change Service Port Numbers* on page 28 for instructions on how to change them.

Service	Port Number
Enterprise Console	8088
Data Integration Server	9080
H2 (SampleData)	9092
Embedded BI Server (Jetty)	10000

How to Change Service Port Numbers

Enterprise Console (Jetty)

Edit the `/pentaho/server/enterprise-console/resource/config/console.properties` file. The port number entries are in the first section at the top of the file.

```
# Management Server Console's Jetty Server Settings

console.start.port.number=8088
console.stop.port.number=8033
```

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 *How to Change the DI Server URL* on page 29 to accommodate for the new port number.

Embedded BI 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.

How to 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 the instructions below. 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.

How to Back Up the Enterprise Repository

Follow the instructions below to create a backup of your PDI enterprise repository.

Note: If you've made any changes to the Pentaho Enterprise Console or DI Server Web application configuration, such as changing the port number or base URL, you will have to modify this procedure to include the entire `/pentaho/server/` directory.

1. Stop the DI Server.

   ```
   /pentaho/server/data-integration-server/stop-pentaho.sh
   ```

2. Create a backup archive or package of the `/pentaho/server/data-integration-server/pentaho-solutions/` directory.

   ```
   tar -cf pdi_backup.tar /pentaho/server/data-integration-server/pentaho-solutions/
   ```

3. Copy the backup archive to removable media or an online backup server.
4. Start the DI Server.

   ```
   /pentaho/server/data-integration-server/start-pentaho.sh
   ```

Your DI Server's stored content, settings, schedules, and user/role information is now backed up.

To restore from this backup, simply unpack it to the same location, overwriting all files that already exist there.

Importing and Exporting Content

You can import and export PDI content to and from an enterprise repository by using PDI's built-in functions, explained in the subsections below.

Note: Among other purposes, these procedures are useful for backing up and restoring content in the enterprise 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 Back Up the Enterprise Repository* on page 30 instead.

Importing Content Into a Pentaho Enterprise Repository

You must be logged into the Enterprise Repository in Spoon.

Follow the instructions below to import the Enterprise Repository.

1. In Spoon, go to **Tools** -> **Repository** -> **Import Repository**.
2. Locate the export (XML) file that contains the enterprise 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.

Using 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 PDI content you're importing. Pentaho provides a sample file called **import-rules.xml**, included with the standard PDI 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 **user**.
dir	The directory in the repository that you want to copy the content to.
limitdir	**Optional.** 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.

Parameter	Definition/value
comment	The comment that will be set for the new revisions of the imported transformations and jobs.
replace	Set to **Y** to replace existing transformations and jobs in the repository. Default value is **N**.
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"
```

Exporting Content From a Pentaho Enterprise Repository

You must be logged into the Enterprise Repository through Spoon.

Follow the instructions below to export the Enterprise Repository.

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 PDI content you selected. It is possible to break it up into individual KTR and KJB files by hand or through a transformation.

Logging and Monitoring

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

How to Enable Logging

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

1. Create a database or table space called **pdi_logging**.

   If you don't have a database set aside specifically for logging and other administrative tasks, you can use the SampleData H2 database service included with PDI. H2 does not require you to create a database or table space; if you specify one that does not exist, H2 silently creates it.
2. Start Spoon, and open a transformation or job that you want to enable logging for.
3. Go to the **Edit** menu and select **Settings...**

   The Settings dialogue will appear.
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 will appear.
7. Enter in your database connection details, then click **Test** to ensure that they are correct. Click **OK** when you're done.

   If you are using the included H2 instance, it is running on **localhost** on port **9092**. Use the **hibuser** username with the **password** password.
8. Look through the list of fields to log, and ensure that the fields you are interested in are checked.

   Warning: Monitoring the **LOG_FIELD** field can negatively impact BI 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 Enterprise Console.

Logging is now enabled for the job or transformation in question.

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** Don't 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

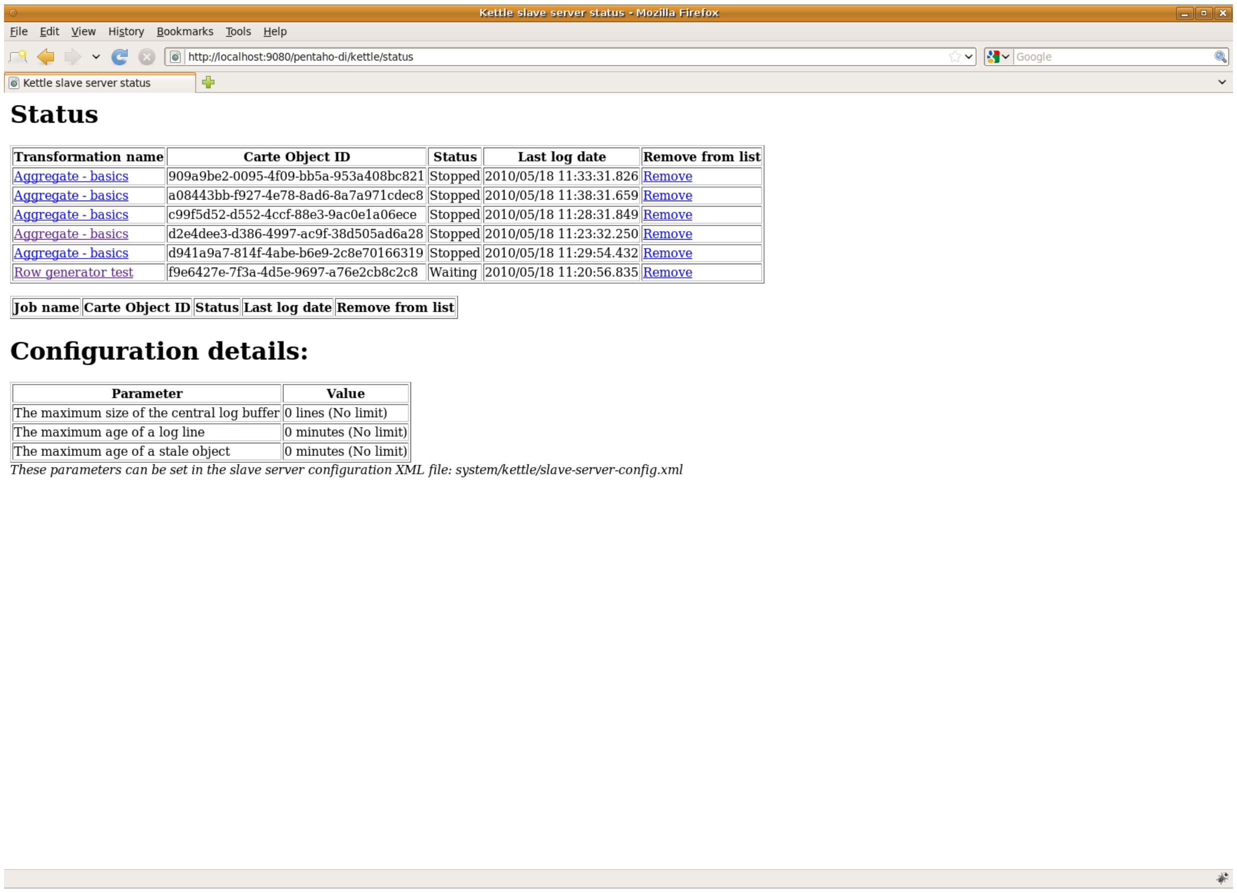
If the **Enable time** option is enabled, all lines in the logging will be preceded by the time of day.

Monitoring 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 DI Server (change the hostname and port to match your configuration):

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

If you aren't yet logged into the DI Server, you'll be redirected to the login page before you can continue. Once logged in, the Kettle Status page should look something like this:


Kettle slave server status - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:9080/pentaho-di/kettle/status
Google
Kettle slave server status

Status

Transformation name	Carte Object ID	Status	Last log date	Remove from list
Aggregate - basics	909a9be2-0095-4f09-bb5a-953a408bc821	Stopped	2010/05/18 11:33:31.826	Remove
Aggregate - basics	a08443bb-f927-4e78-8ad6-8a7a971cdec8	Stopped	2010/05/18 11:38:31.659	Remove
Aggregate - basics	c99f5d52-d552-4ccf-88e3-9ac0e1a06ece	Stopped	2010/05/18 11:28:31.849	Remove
Aggregate - basics	d2e4dee3-d386-4997-ac9f-38d505ad6a28	Stopped	2010/05/18 11:23:32.250	Remove
Aggregate - basics	d941a9a7-814f-4abe-b6e9-2c8e70166319	Stopped	2010/05/18 11:29:54.432	Remove
Row generator test	f9e6427e-7f3a-4d5e-9697-a76e2cb8c2c8	Waiting	2010/05/18 11:20:56.835	Remove

Job name	Carte Object ID	Status	Last log date	Remove from list

Configuration details:

Parameter	Value
The maximum size of the central log buffer	0 lines (No limit)
The maximum age of a log line	0 minutes (No limit)
The maximum age of a stale object	0 minutes (No limit)

These parameters can be set in the slave server configuration XML file: system/kettle/slave-server-config.xml

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* on page 34 for more information on what these settings do and how you can modify them.

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

slave-server-config.xml

Remote execution and logging -- 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. The three configurable options are explained below.

Note: Your DI Server must be stopped in order to make modifications to slave-server-config.xml.

Property	Values	Description
max_log_lines	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.
max_log_timeout_minutes	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.
object_timeout_minutes	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_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>
```

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, you should probably connect it to the Enterprise Console, BA Server, and DI Server and use that instead of implementing the solution below.

Enterprise Console and the BA and DI 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 the instructions below to modify the settings so that Pentaho server log files are rotated and compressed.

1. Stop all relevant Pentaho servers -- BA Server, DI Server, Pentaho Enterprise Console.
2. Download a Zip archive of the Apache Extras Companion for log4j package: *http://logging.apache.org/log4j/companions/extras/*.
3. Unpack the **apache-log4j-extras** JAR file from the Zip archive, and copy it to the following locations:
   - **BA Server:** `/tomcat/webapps/pentaho/WEB-INF/lib/`
   - **DI Server:** `/tomcat/webapps/pentaho-di/WEB-INF/lib/`
   - **Enterprise Console:** `/enterprise-console/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/`
   - **Enterprise Console:** `/enterprise-console/resource/config/`
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 below:

   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.
            Enterprise console would be "server.log" -->
       <param name="File" value="../logs/pentaho.log" />
       <param name="Append" value="false" />
       <rollingPolicy class="org.apache.log4j.rolling.TimeBasedRollingPolicy">
           <!-- See javadoc for TimeBasedRollingPolicy -->
           <!-- Change this value to "server.%d.log.gz" for PEC -->
           <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.
            Enterprise console would be "server.log" -->
       <param name="File" value="../logs/pentaho.log" />
       <param name="Append" value="false" />
       <rollingPolicy class="org.apache.log4j.rolling.FixedWindowRollingPolicy">
           <!-- Change this value to "server.%i.log.gz" for PEC -->
           <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 now have an independent log rotation system in place for all modified Pentaho servers.

PDI Operations Mart

The PDI Operations Mart enables you to collect and query PDI log data, as well as use Pentaho Analyzer or Interactive Reporting to examine log data in reports, charts, or dashboards.

Description

The PDI Operations Mart is a centralized data mart that stores PDI 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 PDI Operations Mart database. Pentaho provides SQL scripts to create these tables for MySQL, Oracle, and PostgresSQL databases. A PDI job populates the time and date dimensions. To create an operations mart you need Pentaho Business Analytics Enterprise Edition.

Getting Started

These procedures describe how to install, set up, and configure a PDI Operations Mart to collect ETL logging information.

Installing the PDI Operations Mart

Setting Up Database Connections for the PDI Operations Mart

Creating the PDI Operations Mart

Configuring ETL Logging Data Collection

Populating the PDI Operations Mart

Setting Up and Distributing the Data Models

Loading the Sample PDI Operations Mart

Installing the PDI Operations Mart

1. Download the PDI Operations Mart archive from *https://support.pentaho.com/entries/22157752-pdi-operations-mart-since-pdi-version-4-4* .

2. Expand the archive to a temporary location.
Archive folder strucfture

- `pdi-operations-mart/`
  - `DDL/`
  - `doc/`
  - `etl/`
  - `models/`
  - `samples/`
    - `dashboards/`
    - `datamart/`
    - `jobs_transformations/`
    - `reports/`

3. On the machine that runs the solution repository associated with the DI Server on which you will schedule and run the ETLjob, create a folder such as `pdi_operations_mart/public`. 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 offline database that contains the PDI Operations Mart database objects.

Your PDI Operations Mart is now ready to be created. See *Creating the PDI Operations Mart* for procedures to create the PDI Operations Mart.

Setting 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 PDI 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 offline Operations Mart database. The connection information needs to match your databases. This procedure explains how to define your database connections.

1. From within PDI/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 fetchs 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 PDI Operations Mart are defined.

Creating the PDI 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, Postgres, 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 PDI/Spoon, run the job named `Fill in DIM_DATE and DIM_TIME`.
   The job completes without error, which indicates the PDI Operations Mart was created.

Configuring Logging Data Collection

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

It is a best practie to 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 PDI/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_OUPUT`, 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 PDI Operations Mart is configured to collect ETL logging data.

Updating the PDI Operations Mart

Once the Operations Mart has been *created* and you configured which *logging data* you would like to collect, you can update the data periodically to monitor the latest performance of your ETL operations. For instance, you might update the executor and dimensions tables if you

- Add, remove, or change an Operations Mart database connection
- Add, remove, or change a transformation step or job entry

However, if you only want to update the logging data collected in the Operations Mart, you only need to populate the Operations Mart with the latest data in the log tables.

Perform steps 1 and 2 if you change transformations, jobs, or Operations Mart database connections. This updates the executor and log table dimensions before populating the Operations Mart tables with the latest logging data.

Perform step 1 only if you only want to update the Operations Mart with the latest logging data. You do not need to update the executor and log table dimensions, which takes longer than just populating with the latest logging data.

1. Update the executor and logging dimension tables by running the **Update Executor and Log Table Dimensions** transformation.
2. Populate the Operations Mart with the latest logging information by running the **Update Logging Datamart** job. The fact table and remaining dimensions are populated using data from the online logging database.

The job completes without errors, which indicates your Operations Mart is populated with the latest logging data.

Loading the Sample PDI 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 PDI/Spoon.
3. From within PDI/Spoon, execute the **Fill_datamart_with_sample_data.kjb** job.

The sample data is loaded into the PDI Operations Mart.

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 PDI Operations Mart under the BA Server.

1. From within the Pentaho 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.

This procedure can be repeated for loading sample charts or dashboards by substituting either for the word *reports* in step 2.

Setting Up and Distributing the Data Models

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

1. Place the `models/metadata.xmi` and the `models/PDI_Operations_Mart_<dbname>.mondrian.xml`, where <dbname> is the name of the database containing the PDI 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 `metadata.xmi` 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 `metadata.xmi` in this location, you need to rename the file to `PDI_Operations_Mart.xmi`. The same applies accordingly to the Mondrian schema.
2. Rename the <filepath>PDI_Operations_Mart_<dbname>.mondrian.xml</filepath> file to `PDI_Operations_Mart.mondrian.xml`.
3. From within the Pentaho Enterprise Console, create a new database connection. Name it `PDI_Operations_Mart`. Then set the connection information to the PDI Operations Mart database location. You can copy the needed URL from the database connecting within PDI/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 datasource `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. From the Pentaho Enterprise Console, restart the BA Server.
7. From the Pentaho User Console, refresh all caches and settings.

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

Creating Charts, Reports, and Dashboards Using PDI Operations Mart Data

Once you have created and populated your PDI Operations Mart with log data, the featueres of Pentaho User Console (PUC) enable you to examine this data and create reports, charts, and dashboards. Pentaho provides 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 the *Pentaho User Console User Guide*.

Creating ETL Logging Reports

1. Open a new report in Pentaho 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/metadata.xmi`.
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 Kettle Type 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.

Creating 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: `="" & (ROWCOUNT([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.

Creating ETL Logging Dashboards

This procedurs 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 the *Pentaho User Console User Guide* for more information about creating and customizing dashboards.

1. From with Pentaho 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 now have a dashboard that displays log data created using the PDI Operations Mart.

PDI Operations Mart Logging Tables Status Descriptions

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.

PDI Operation Mart Logging Dimensions and Metrics

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.

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.

Field Name	Description
`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
`executor_source`	The source location (either file- or repository-relative) for the executor
`* executor_environment`	File server, repository name, related to the executor_source. **Reserved for future use.*
`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.

Field Name	Description
`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
`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.
`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 dimenstion 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.). **Reserved for future use.*

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).

Field Name	Description
`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

Field Name	Description
`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.

PDI Operations Mart Best Practices

Cleaning 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 Pentaho Data Integration User Guide, *Scheduling Transformations and Jobs From Spoon*.

Using PDI Data Sources in Action Sequences

If you have any action sequences that rely on Pentaho Data Integration (PDI) data sources that are stored in an enterprise repository, you must make a configuration change in order to run them.

Create a `.kettle` directory in the home directory of the user account that runs the BI Server, and copy the **repositories.xml** file from your local PDI configuration directory to the new one you just created on the BI Server machine.

You must also edit the `/pentaho-solutions/system/kettle/settings.xml` file and put in your PDI enterprise repository information.

Once these changes have been made, restart the BI Server. When it comes back up, the **Use Kettle Repository** function in Pentaho Design Studio should properly connect to the DI Server.

Configuring Your Big Data Environment

This section covers configuring PDI to communicate with Hadoop distributions other than the default configuration, Apache Hadoop 0.20.X. A list of supported big data technologies can be found in the *PDI Installation Guide, Compatability Matrix*.

Setting the Active Hadoop Configuration

Hadoop configurations within PDI are collections of the Hadoop libraries required to communicate with a specific version of Hadoop and Hadoop-related tools, such as Hive, HBase, Sqoop, or Pig.

The Kettle client comes pre-configured for Apache Hadoop 0.20.X, no further configuration is required in order to use PDI with this version of Hadoop.

Locating Which Version of Hadoop Is In Use

The Hadoop distribution configuration can be found at this location: `plugins/pentaho-big-data-plugin/plugin.properties`. In that file, locate this statement.

```
# The Hadoop Configuration to use when communicating with a Hadoop cluster.
# This is used for all Hadoop client tools including HDFS, Hive, HBase, and Sqoop.
active.hadoop.configuration=hadoop-20
```

The property `active.hadoop.configuration` sets the distribution of Hadoop to use when communicating with a Hadoop cluster. This is the property to update if you are using a version other than Apache Hadoop 0.20.X.

Configuring for Cloudera

To communicate with Cloudera, you must change which version of Hadoop to use when communicating with a Hadoop cluster.

Within the file `plugins/pentaho-big-data-plugin/plugin.properties`, update the `active.hadoop.configuration` property to match your distribution of Cloudera.

For CDH3u4, update the `active.hadoop.configuration` property to `cdh3u4`.

For CDH4, update the `active.hadoop.configuration` property to `cdh4`.

Configuring for MapR

To communicate with MapR, the MapR client tools must be installed on the local machine.

For All Operating Systems

1. Within the file `plugins/pentaho-big-data-plugin/plugin.properties`, update the `active.hadoop.configuration` property to `mapr`.

For Windows

2. Within the file `pentaho\design-tools\data-integration\plugins\pentaho-big-data-plugin\hadoop-configurations\mapr\config.properties`.
   a) Remove the existing `classpath` and `library.path` properties by deleting

   ```
   classpath=/opt/mapr/conf,/opt/mapr/hadoop/hadoop-0.20.2/lib
   ```

   ```
   library.path=/opt/mapr/lib
   ```

   b) Uncomment the Windows-specific `classpath` and `library.path` properties and edit them to match your local MapR client tools installation directory. Once completed, the properties should look like this.

   ```
   classpath=file:///C:/opt/mapr/conf,file:///C:/opt/mapr/mapr-
   client-1.2.9.14720GA-0.amd64/hadoop/hadoop-0.20.2/lib
   ```

   ```
   library.path=C:\\opt\\mapr\\mapr-client-1.2.9.14720GA-0.amd64\\lib
   ```

Creating a New Hadoop Configuration

If you have a Hadoop distribution not supported by Pentaho, or you have modified your Hadoop Installation in such a way that it is no longer compatible with Pentaho, you may need to create a new Hadoop configuration.

Changing which version of Hadoop PDI can communicate with requires you to swap the appropriate `jar` files within the plugin directory and then update the `plugin.properties` file.

1. Identify which Hadoop configuration most closely matches the version of Hadoop you want to communicate with. If you compare the default configurations included the differences are apparent.
2. Copy the `jar` files for your specified Hadoop version.
3. Paste the `jar` files into the `lib/` directory.
4. Change the `active.hadoop.configuration=` property in the `plugins/pentaho-big-dataplugin/hadoop-configurations/plugin.properties` file to match your specific Hadoop configuration. This property configures which distribution of Hadoop to use when communicating with a Hadoop cluster. Update this property if you are using a version other than the default Hadoop version.

Troubleshooting

This section contains known problems and solutions relating to the procedures covered in this guide.

I don't know what the default login is for the DI Server, Enterprise Console, and/or Carte

For the DI Server administrator, it's username **admin** and password **secret**.

For Enterprise Console administrator, it's username **admin** and password **password**.

For Carte, it's username **cluster** and password **cluster**.

Be sure to change these to new values in your production environment.

Note: DI Server users are not the same as BI Server users.

Jobs scheduled on the DI Server cannot execute a transformation on a remote Carte server

You may see an error lie 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* on page 26

PDI Transformation Logging Doesn't Show In PEC

If you've enabled logging for a PDI transformation, then check the Pentaho Enterprise Console for log information, you may not find it if you didn't select all available fields, including **LOG_FIELD**, when you chose transformation fields.

To fix this, reconfigure the transformation logging options to include all available fields.

Troubleshooting a Corrupted DI Server Repository

If the DI Server's enterprise repository becomes corrupt, it will be unresponsive, content may be missing or inaccessible, and an error message similar to this will appear in the `/data-integration-server/tomcat/logs/catalina.out` log file:

```
ERROR [ConnectionRecoveryManager] could not execute statement, reason: File corrupted
 while reading record: "page[48970] data leaf table:8 entries:1 parent:49157 keys:
[118547] offsets:[737]". Possible solution: use the recovery tool [90030-131], state/
code: 90030/90030
```

If this happens, shut down the DI Server and restore your enterprise repository from a recent backup.

If you do not have a viable backup, you may be able to minimize data loss by identifying the exact file that is corrupt. To do this, enable debug logging by adding the following XML snippet above the **<root>** element in the `/WEB-INF/classes/log4j.xml` inside your deployed pentaho.war:

```
<category name="org.pentaho.platform">
     <priority value="DEBUG"/>
</category>
```

Restart the DI Server and retry the action that caused the original error. If it occurs again, shut down the DI Server and open the **catalina.out** log file in Tomcat. Look for the last line that appears before the error; it usually contains the name of the file that has been damaged. When you are finished investigating the data corruption, remove the extra logging capabilities so that your DI Server log files don't become large and unmanageable.

```
reading file with id 'xyz' and path '/public/a.txt'
```

You can also try to recover your PDI data from the damaged database by using a recovery program, as explained in *Using the H2 Database Recovery Tool* on page 54.

Note: If the database has become corrupt, the damaged rows will not be exported with the recovery tool, and whatever information they contained will be lost.

Using the H2 Database Recovery Tool

The DI Server includes a third-party H2 database recovery tool that enables you to extract raw data from your PDI enterprise repository. This is primarily useful in situations where the repository has become corrupt, and you don't have any relevant backups to restore from.

Note: If the database has become corrupt, the corrupted rows will not be exported. Any information contained in corrupted rows is unrecoverable through this method.

The recovery tool is a JAR that must be run via the Java command. The output is a SQL dump that you can then attempt to import after you've re-initialized your DI Server database.

You can read more about the recovery tool on the H2 Web site: *http://www.h2database.com/html/advanced.html#using_recover_tool*.

Follow the directions below to use the recovery tool.

1. Open a terminal on (or establish an SSH connection to) your DI Server.
2. Navigate to the `/pentaho-solutions/system/jackrabbit/repository/version/` directory.

   ```
   cd data-integration-server/pentaho-solutions/system/jackrabbit/repository/version/
   ```
3. Run the **h2-1.2.131.jar** H2 database JAR with the recovery tool option.

   ```
   java -cp h2-1.2.131.jar org.h2.tools.Recover
   ```
4. Create a directory to move your old database files to.

   ```
   mkdir old
   ```
5. Move the old database files to the directory you just created.

   ```
   mv db.h2.db db.trace.db old
   ```
6. Re-initialize the repository with the **RunScript** option, using the salvaged SQL dump as the source.

   ```
   java -cp h2-1.2.131.jar org.h2.tools.RunScript -url jdbc:h2:./db -user sa -script
    db.h2.sql
   ```
7. The backup directory you created earlier (**old** in the above example) can be removed after you're sure that you don't need the corrupted database files anymore. However, it's a good idea to keep this around just in case you need it later.

You've successfully extracted all of the usable data from your corrupt enterprise repository, removed the damaged database files, and re-initialized the repository with the salvaged data.

Start the DI Server and check for further errors. If repository errors persist, contact Pentaho support and request developer assistance.

DI Server fails to start (IOException: Premature end of file)

If the DI Server fails to start, and you see an exception error like the one printed in the example, then follow the below directions to fix the **custom_nodetypes.xml** file.

1. Stop the DI Server.
2. Create a backup copy of the `/data-integration-server/pentaho-solutions/system/jackrabbit/` directory.
3. Delete the `/data-integration-server/pentaho-solutions/system/jackrabbit/` directory.
4. Create a new empty `/data-integration-server/pentaho-solutions/system/jackrabbit/` directory.
5. Create a new empty `/data-integration-server/pentaho-solutions/system/jackrabbit/repository/` directory.
6. Copy the **repository.xml** file from the directory you just backed up to the new jackrabbit directory.

   ```
   cp ../backup_jackrabbit/repository.xml ./
   ```
7. Start the DI Server
8. Wait until the DI Server has fully started and initialized.
9. Stop the DI Server.
10. Copy the `/data-integration-server/pentaho-solutions/system/jackrabbit/repository/repository/nodetypes/custom_nodetypes.xml` file to the same location in the backup directory you created earlier.

    ```
    cp custom_nodetypes.xml ../../../../backup_jackrabbit/repository/repository/
    nodetypes/
    ```
11. Delete the newly-created `/data-integration-server/pentaho-solutions/system/jackrabbit/` directory.
12. Move the backup copy of the jackrabbit directory to its original location.

    ```
    mv backup_jackrabbit /home/pentaho/pentaho/server/data-integration-server/pentaho-
    solutions/system/jackrabbit/
    ```
13. Start the DI Server

The DI Server should now start normally.

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.

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 enterprise repositories
.spoonrc	User interface settings, including the last opened transformation/job
.languageChoice	Default language for the PDI client tool

Changing 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`)

Since this is user-specific, there will be a different .kettle directory (and thus 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
```

BI Server deployments that run PDI content

If you followed a manual deployment or archive package installation path, you can simply set a system environment variable as explained above, but it must be declared before the BI 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 BI Server and set the flag inline, as in this example from the **start-pentaho.sh** script:

```
export CATALINA_OPTS="--Xms256m -Xmx768m -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 BI Server, then you must modify the Java options flag that runs the BI 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
```

The **DI Server** is similarly modified, the only change being the service name:

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